

BBBBBBBBBBBBBB		AAAAAAAAAA	CCCCCCCCCCCC	KKK	KKK	UUU	UUU	PPPPPPPPPPPP
BBBBBBBBBBBBBB		AAAAAAAAAA	CCCCCCCCCCCC	KKK	KKK	UUU	UUU	PPPPPPPPPPPP
BBBBBBBBBBBBBB		AAAAAAAAAA	CCCCCCCCCCCC	KKK	KKK	UUU	UUU	PPPPPPPPPPPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBBBBBBBBBBBBB		AAA	AAA	CCC	KKKKKKKKKK	UUU	UUU	PPPPPPPPPPPP
BBBBBBBBBBBBBB		AAA	AAA	CCC	KKKKKKKKKK	UUU	UUU	PPPPPPPPPPPP
BBBBBBBBBBBBBB		AAA	AAA	CCC	KKKKKKKKKK	UUU	UUU	PPPPPPPPPPPP
BBB	BBB	AAAAAAAAAAAAAAAA	CCC	KKK	KKK	UUU	UUU	PPP
BBB	BBB	AAAAAAAAAAAAAAAA	CCC	KKK	KKK	UUU	UUU	PPP
BBB	BBB	AAAAAAAAAAAAAAAA	CCC	KKK	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBB	BBB	AAA	AAA	CCC	KKK	UUU	UUU	PPP
BBBBBBBBBBBBBB		AAA	AAA	CCCCCCCCCCCC	KKK	KKK	UUUUUUUUUUUUUUUU	PPP
BBBBBBBBBBBBBB		AAA	AAA	CCCCCCCCCCCC	KKK	KKK	UUUUUUUUUUUUUUUU	PPP
BBBBBBBBBBBBBB		AAA	AAA	CCCCCCCCCCCC	KKK	KKK	UUUUUUUUUUUUUUUU	PPP

```
FFFFFFFFF      AAAAAA      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      AAAAAA      NN      NN
FFFFFFFFF      AAAAAA      SSSSSSSS      TTTTTTTTTT      SSSSSSSS      CCCCCCCC      AAAAAA      NN      NN
FF          AA      AA      SS          TT          SS          CC          AA      AA      NN      NN
FF          AA      AA      SS          TT          SS          CC          AA      AA      NN      NN
FF          AA      AA      SS          TT          SS          CC          AA      AA      NNNN      NN
FF          AA      AA      SS          TT          SS          CC          AA      AA      NNNN      NN
FFFFFFFFF      AA      AA      SSSSSS      TT          SSSSSS      CC          AA      AA      NN      NN
FFFFFFFFF      AA      AA      SSSSSS      TT          SSSSSS      CC          AA      AA      NN      NN
FF          AAAAAAAAAA      SS          TT          SS          CC          AAAAAAAAAA      NN      NNNN
FF          AAAAAAAAAA      SS          TT          SS          CC          AAAAAAAAAA      NN      NNNN
FF          AA      AA      SS          TT          SS          CC          AA      AA      NN      NN
FF          AA      AA      SSSSSSSS      TT          SSSSSSSS      CCCCCCCC      AA      AA      NN      NN
FF          AA      AA      SSSSSSSS      TT          SSSSSSSS      CCCCCCCC      AA      AA      NN      NN
                                     ....
                                     ....
                                     ....
                                     ....

LLLL          IIIIII      SSSSSSSS
LLLL          IIIIII      SSSSSSSS
LLLL          II      SS
LLLL          II      SS
LLLL          II      SS
LLLL          II      SS
LLLL          II      SSSSSS
LLLL          II      SSSSSS
LLLL          II      SS
LLLL          II      SS
LLLL          II      SS
LLLL          II      SS
LLLLLLLLLLLL  IIIIII      SSSSSSSS
LLLLLLLLLLLL  IIIIII      SSSSSSSS
```



```
0001 0 MODULE FASTSCAN (XTITLE 'Fast file scan'
0002 0 IDENT = 'V04-000'
0003 0 ) =
0004 1 BEGIN
0005 1
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *
0028 1 *****
0029 1
0030 1
0031 1 ++
0032 1 FACILITY:
0033 1 Backup/Restore
0034 1
0035 1 ABSTRACT:
0036 1 This module contains the fast file scan routines.
0037 1
0038 1 ENVIRONMENT:
0039 1 VAX/VMS user mode.
0040 1 --
0041 1
0042 1 AUTHOR: M. Jack, CREATION DATE: 19-Nov-1980
0043 1
0044 1 MODIFIED BY:
0045 1
0046 1 V03-015 LMP0272 L. Mark Pilant, 6-Jul-1984 8:41
0047 1 Modify BACKUP to always use a full length FIB.
0048 1
0049 1 V03-014 LY0499 Larry Yetto 25-JUN-1984 12:36
0050 1 Modify the concealed root directory code to properly deal
0051 1 with [000000]
0052 1
0053 1 V03-013 ACG0381 Andrew C. Goldstein, 9-Dec-1983 19:43
0054 1 Fix byte count of initial MFD file name string
0055 1 (broken by ACG0366)
0056 1
0057 1 V03-012 ACG0366 Andrew C. Goldstein, 11-Oct-1983 18:55
```



58	0058	1	Fix rooted directory handling to track \$PARSE changes
59	0059	1	
60	0060	1	V03-011 ACG0325 Andrew C. Goldstein, 4-Apr-1983 15:51
61	0061	1	Fix header area length validation
62	0062	1	
63	0063	1	V03-010 MLJ0105 Martin L. Jack, 14-Feb-1983 17:07
64	0064	1	Correct error in execute-only "no such file" path.
65	0065	1	
66	0066	1	V03-009 ACG0313 Andrew C. Goldstein, 12-Feb-1983 16:12
67	0067	1	Add routine subtitles
68	0068	1	
69	0069	1	V03-008 MLJ0104 Martin L. Jack, 24-Jan-1983 18:51
70	0070	1	Correct access control bits used for index file under
71	0071	1	/IGNORE=INTERLOCK. Allow access to execute-only directories
72	0072	1	via ACP lookup. Change bootblock handling to allow copying RSX
73	0073	1	system disks. Tighten V03-006 to save scanned but non-selected
74	0074	1	directories only if the selection file specification is "*.*;*"
75	0075	1	and the directory terminator is "*".
76	0076	1	
77	0077	1	V03-007 MLJ0101 Martin L. Jack, 13-Nov-1982 19:46
78	0078	1	Correct ODS-1 latest-version selection.
79	0079	1	
80	0080	1	V03-006 MLJ48949 Martin L. Jack, 6-Sep-1982 17:17
81	0081	1	Save scanned but non-selected directories only if the selection
82	0082	1	file specification is "*.*;*". This saves space in cases where
83	0083	1	an incremental restore is meaningless anyway.
84	0084	1	
85	0085	1	V03-005 MLJ0096 Martin L. Jack, 24-Aug-1982 15:50
86	0086	1	Correct /IMAGE file number conflict bug.
87	0087	1	
88	0088	1	V03-004 MLJ48500 Martin L. Jack, 18-Aug-1982 10:23
89	0089	1	Correct bad test that caused scanned but non-selected
90	0090	1	directories to be copied in a disk to disk operation. Also
91	0091	1	add a test to avoid extra accesses to directories.
92	0092	1	
93	0093	1	V03-003 MLJ0087 Martin L. Jack, 08-Apr-1982 16:45
94	0094	1	Do not truncate the index file for ODS-1 version 2 because it
95	0095	1	cannot be re-extended to multi-header dynamically. Fix setting
96	0096	1	of DIR_STATUS.
97	0097	1	
98	0098	1	V03-002 MLJ0083 Martin L. Jack, 22-Mar-1982 13:25
99	0099	1	Inhibit saving of scanned but not selected directories under
100	0100	1	/INTERCHANGE, to save space on distribution media.
101	0101	1	
102	0102	1	V03-001 MLJ0082 Martin L. Jack, 16-Mar-1982 18:42
103	0103	1	Initialize DIR STATUS before index file scan to avoid incorrect
104	0104	1	results from SELECT_INPUT_FILE.
105	0105	1	
106	0106	1	V02-011 MLJ0078 Martin L. Jack, 10-Feb-1982 15:30
107	0107	1	Correct error in exclusion of aliased files introduced by
108	0108	1	V02-007.
109	0109	1	
110	0110	1	V02-010 MLJ0077 Martin L. Jack, 8-Feb-1982 15:11
111	0111	1	Implement negative version numbers.
112	0112	1	
113	0113	1	V02-009 MLJ0075 Martin L. Jack, 28-Jan-1982 20:16
114	0114	1	Implement DIR_VERLIM and VERLIMIT attributes to support version



115	0115	1	!	limit handling. Add FIB\$V_NORECORD to file accesses to support
116	0116	1	!	file expiration handling.
117	0117	1	!	
118	0118	1	!	V02-008 MLJ0063 Martin L. Jack, 22-Dec-1981 4:20
119	0119	1	!	Support rooted directories.
120	0120	1	!	
121	0121	1	!	V02-007 MLJ0062 Martin L. Jack, 4-Dec-1981 14:33
122	0122	1	!	Implement /INCREMENTAL.
123	0123	1	!	
124	0124	1	!	V02-006 MLJ0054 Martin L. Jack, 15-Oct-1981 19:26
125	0125	1	!	Support segmented directory records. Combine routines
126	0126	1	!	ODS1_VOLUME_ATTRIBUTES and ODS2_VOLUME_ATTRIBUTES. Remove
127	0127	1	!	COM_IMP_NOBACK, as this must be computed on every file.
128	0128	1	!	Implement /VOLUME. Implement /IGNORE=INTERLOCK. Integrate
129	0129	1	!	GET_VM and FREE_VM jacket routines.
130	0130	1	!	
131	0131	1	!	V02-005 MLJ0037 Martin L. Jack, 29-Aug-1981 16:04
132	0132	1	!	Correct file selection with /IMAGE.
133	0133	1	!	
134	0134	1	!	V02-004 MLJ0036 Martin L. Jack, 28-Aug-1981 17:55
135	0135	1	!	Reimplement file scan.
136	0136	1	!	
137	0137	1	!	V02-003 MLJ0025 Martin L. Jack, 8-May-1981 11:25
138	0138	1	!	Implement latest-version selection. Make /RECORD restartable.
139	0139	1	!	
140	0140	1	!	V02-002 MLJ0018 Martin L. Jack, 7-Apr-1981 21:08
141	0141	1	!	Restore expanded string length in input NAM block.
142	0142	1	!	
143	0143	1	!	V02-001 MLJ0010 Martin L. Jack, 25-Mar-1981 16:33
144	0144	1	!	Reorganize global storage. Reorganize so that index file
145	0145	1	!	processing occurs on one volume at a time to minimize global
146	0146	1	!	storage requirement. Incorporate "slow" file scan into fast
147	0147	1	!	file scan so that BACKUP, not RMS, maintains directory context.
148	0148	1	!	Add attributes check on directories so that spurious
149	0149	1	!	"illegal format" errors are not produced.
150	0150	1	!	
151	0151	1	! **	

```
153 0152 1 REQUIRE 'SRC$:COMMON';
154 1258 1 LIBRARY 'SYSS$LIBRARY:LIB';
155 1259 1
156 1260 1
157 1261 1 FORWARD ROUTINE
158 1262 1 FAST_FILE_SCAN: NOVALUE, ! Scan with index file
159 1263 1 SLOW_FILE_SCAN: NOVALUE, ! Scan without index file
160 1264 1 READ_HOMEBLOCK: NOVALUE, ! Read and check home block
161 1265 1 VERIFY_HEADER, ! Check a file header
162 1266 1 PROCESS_FILE: NOVALUE, ! Process one file
163 1267 1 DIR_SCAN: NOVALUE, ! Driver for directory scan
164 1268 1 INIT_DIR_SCAN: NOVALUE, ! Initialize directory scan
165 1269 1 RESET_DIR_SPEC: NOVALUE, ! Reset selection filespec
166 1270 1 FIND_NEXT, ! Scan for next matching file
167 1271 1 FREE_DIR_DATA: NOVALUE, ! Free directory scan context
168 1272 1
169 1273 1
170 1274 1 EXTERNAL ROUTINE
171 1275 1 LIB$EXTRACT_CONCEALED: ADDRESSING_MODE(GENERAL),
172 1276 1 ! Parse concealed device/root directory
173 1277 1 CHECKSUM, ! Compute file header checksum
174 1278 1 CHECKSUM2, ! Compute home block checksum
175 1279 1 GEN_FID_RECORD: NOVALUE, ! Write FID record
176 1280 1 INIT_NAMEBLOCK: NOVALUE, ! Initialize extended name block fields
177 1281 1 VOLUME_ATTRIBUTES: ! Write volume attribute record
178 1282 1 NOVALUE,
179 1283 1 SAVE_ONE_FILE, ! Routine to copy one file
180 1284 1 FILE_ERROR: NOVALUE, ! Signal file-related error
181 1285 1 INIT_ATTR: NOVALUE, ! Initialize attributes area
182 1286 1 LEFT_ONE, ! Find leftmost one bit in a word
183 1287 1 MAKE_STRING, ! Convert ODS-1 filename to ASCII
184 1288 1 INIT_SEL_INFO: NOVALUE, ! Initialize selection information
185 1289 1 MATCH_DIRECTORY, ! Match directory specification
186 1290 1 MATCH_FILENAME, ! Match file name, type, and version
187 1291 1 SELECT_INPUT_FILE, ! Select input file based on qualifiers
188 1292 1 TERMINATE_SCAN, ! Test termination of directory scan
189 1293 1 FREE_VM: NOVALUE, ! Free virtual memory
190 1294 1 GET_VM, ! Get virtual memory
191 1295 1 GET_ZERO_VM, ! Get and zero virtual memory
192 1296 1 LIB$SIGNAL: ADDRESSING_MODE(GENERAL);
193 1297 1 ! Signal a condition
194 1298 1
195 1299 1
196 1300 1 EXTERNAL LITERAL
197 1301 1 BACKUP$MAXVOLS,
198 1302 1 BACKUP$PROCINDEX,
199 1303 1 BACKUP$OPENDIR,
200 1304 1 BACKUP$OPENIN,
201 1305 1 BACKUP$READDIR,
202 1306 1 BACKUP$BADDIR,
203 1307 1 BACKUP$NOSUCHRVN,
204 1308 1 LIB$INVFILSPEC;
205 1309 1
206 1310 1
207 1311 1 G$DEFINE(); ! Define global common area
208 1312 1
209 1313 1
```



FASTSCAN  
V04-000

Fast file scan

B 3  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 5  
(2)

:	210	1314	1	LITERAL			
:	211	1315	1		MAX_VOLUMES=	255,	! Largest volume set
:	212	1316	1		DIR_BUF_COUNT=	16,	! Size of directory buffer
:	213	1317	1		INDEX_BUF_COUNT=	64;	! Size of index file buffer
:	214	1318	1				
:	215	1319	1				
:	216	1320	1	BIND			
:	217	1321	1		MFD=	UPLIT BYTE (%ASCIC '000000');	
:	218	1322	1				
:	219	1323	1				
:	220	1324	1	BUILTIN			
:	221	1325	1		TESTBITCC,		
:	222	1326	1		TESTBITSC,		
:	223	1327	1		CALLG,		
:	224	1328	1		ROT;		



```

: 226 1329 1 %SBTTL 'FAST FILE SCAN - fast file scan main routine'
: 227 1330 1 GLOBAL ROUTINE FAST_FILE_SCAN: NOVALUE=
: 228 1331 1
: 229 1332 1 !++
: 230 1333 1
: 231 1334 1 FUNCTIONAL DESCRIPTION:
: 232 1335 1 This routine is the driver for the fast file scan.
: 233 1336 1
: 234 1337 1 INPUT PARAMETERS:
: 235 1338 1 NONE
: 236 1339 1
: 237 1340 1 IMPLICIT INPUTS:
: 238 1341 1 INPUT_FAB - Pointer to current input FAB.
: 239 1342 1 INPUT_NAM - Pointer to current input NAM block.
: 240 1343 1
: 241 1344 1 OUTPUT PARAMETERS:
: 242 1345 1 NONE
: 243 1346 1
: 244 1347 1 IMPLICIT OUTPUTS:
: 245 1348 1 NONE
: 246 1349 1
: 247 1350 1 ROUTINE VALUE:
: 248 1351 1 NONE
: 249 1352 1
: 250 1353 1 SIDE EFFECTS:
: 251 1354 1 NONE
: 252 1355 1
: 253 1356 1 !--
: 254 1357 1
: 255 1358 2 BEGIN
: 256 1359 2 LOCAL
: 257 1360 2
: 258 1361 2 RVN, ! Relative volume number
: 259 1362 2 FIB: BBLOCK[FIB$C_LENGTH], ! FIB
: 260 1363 2 FIB_DESC: VECTOR[2], ! Descriptor for FIB
: 261 1364 2 STATUS, ! Status variable
: 262 1365 2 IOSB: VECTOR[4,WORD]; ! I/O status block
: 263 1366 2
: 264 1367 2 ! Initialize FIB descriptor.
: 265 1368 2
: 266 1369 2 FIB_DESC[0] = FIB$C_LENGTH;
: 267 1370 2 FIB_DESC[1] = FIB;
: 268 1371 2
: 269 1372 2
: 270 1373 2 ! Allocate the general buffer.
: 271 1374 2
: 272 1375 2 COM_FLAGS[COM_DSBL_CHKPT] = TRUE;
: 273 1376 2 FAST_BUFFER_SIZE = 512 * INDEX_BUF_COUNT;
: 274 1377 2 FAST_BUFFER = GET_VM(512 * INDEX_BUF_COUNT);
: 275 1378 2
: 276 1379 2
: 277 1380 2 ! Loop over all volumes in the volume set.
: 278 1381 2
: 279 1382 2 RVN = 1;
: 280 1383 2 DO
: 281 1384 2 BEGIN
: 282 1385 2 LOCAL
```



```
283 1386 3      ATR_DESC:      BBLOCK[12],      ! Attribute list
284 1387 3      STATBLK:      BBLOCK[8],      ! Statistics block
285 1388 3      ACCTL,      ! FIBSM_WRITE or 0
286 1389 3      EOF,      ! Index file EOF position
287 1390 3      CLUSTER,      ! Cluster factor
288 1391 3      BITMAP_OFFSET,      ! Index file bitmap VBN offset
289 1392 3      VBN;      ! Current index file VBN
290 1393 3
291 1394 3
292 1395 3      ! Access the index file on RVN 1.
293 1396 3
294 1397 3      CH$FILL (0, FIB$C_LENGTH, FIB);
295 1398 3      ACCTL = FIBSM_WRITE OR FIBSM_NORECORD;
296 1399 3      FIB[FIB$L_ACCTL] = .ACCTL;
297 1400 3      FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
298 1401 3      FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
299 1402 3      FIB[FIB$W_FID_RVN] = .RVN;
300 1403 3      ATR_DESC[0,0,16,0] = 8;
301 1404 3      ATR_DESC[2,0,16,0] = ATR$C_STATBLK;
302 1405 3      ATR_DESC[4,0,32,0] = STATBLK;
303 1406 3      ATR_DESC[8,0,32,0] = 0;
304 1407 3      STATUS = $QIOW(
305 P 1408 3          FUNC=IOS$ ACCESS OR IOSM_ACCESS,
306 P 1409 3          CHAN=.INPUT_CHAN,
307 P 1410 3          IOSB=IOSB,
308 P 1411 3          P1=FIB_DESC,
309 1412 3          P5=ATR_DESC);
310 1413 3      IF .STATUS THEN STATUS = .IOSB[0];
311 1414 3      IF .STATUS EQL SSS_WRTLOCK
312 1415 3      THEN
313 1416 4          BEGIN
314 1417 4              CH$FILL (0, FIB$C_LENGTH, FIB);
315 1418 4              ACCTL = FIBSM_NORECORD;
316 1419 4              FIB[FIB$L_ACCTL] = FIBSM_NORECORD;
317 1420 4              FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
318 1421 4              FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
319 1422 4              FIB[FIB$W_FID_RVN] = .RVN;
320 P 1423 4              STATUS = $QIOW(
321 P 1424 4                  FUNC=IOS$ ACCESS OR IOSM_ACCESS,
322 P 1425 4                  CHAN=.INPUT_CHAN,
323 P 1426 4                  IOSB=IOSB,
324 P 1427 4                  P1=FIB_DESC,
325 1428 4                  P5=ATR_DESC);
326 1429 4              IF .STATUS THEN STATUS = .IOSB[0];
327 1430 4              END;
328 1431 3      IF NOT .STATUS
329 1432 3      THEN
330 1433 3          SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
331 1434 3
332 1435 3
333 1436 3      ! Read the home block. If RVN 1, establish the size of the volume set
334 1437 3      ! and the structure level.
335 1438 3
336 1439 3      READ_HOMEBLOCK(.RVN, .FAST_BUFFER);
337 1440 3
338 1441 3
339 1442 3      ! Allocate the per-volume data. Each variable between FAST_VOL_BEG and
```

```

340      1443 3      ! FAST_VOL_END is a pointer to a vector that varies with the number of
341      1444 3      ! volumes in the volume set.
342      1445 3      !
343      1446 3      IF .RVN EQL 1
344      1447 3      THEN
345      1448 4          BEGIN
346      1449 4              INCRA A FROM FAST_VOL_BEG TO FAST_VOL_END-%UPVAL BY %UPVAL DO
347      1450 5                  BEGIN
348      1451 5                      .A = GET_ZERO_VM(.COM_1_SETCOUNT*%UPVAL);
349      1452 4                  END;
350      1453 3              END;
351      1454 3
352      1455 3
353      1456 3      ! Initialize information from the home block.
354      1457 3      !
355      1458 3      IF .FAST_STRUCLEV EQL 2
356      1459 3      THEN
357      1460 4          BEGIN
358      1461 4              CLUSTER = .FAST_BUFFER[HM2$W_CLUSTER];
359      1462 4              FAST_IMAP_SIZE[.RVN-1] = .FAST_BUFFER[HM2$W_IBMAPSIZE];
360      1463 4              BITMAP_OFFSET = .CLUSTER*4 + 1;
361      1464 4              FAST_HDR_OFFSET[.RVN-1] = .CLUSTER*4 + .FAST_BUFFER[HM2$W_IBMAPSIZE];
362      1465 4          END
363      1466 3      ELSE
364      1467 4          BEGIN
365      1468 4              CLUSTER = 1;
366      1469 4              FAST_IMAP_SIZE[.RVN-1] = .FAST_BUFFER[HM1$W_IBMAPSIZE];
367      1470 4              BITMAP_OFFSET = 3;
368      1471 4              FAST_HDR_OFFSET[.RVN-1] = 2 + .FAST_BUFFER[HM1$W_IBMAPSIZE];
369      1472 4          END;
370      1473 3
371      1474 3
372      1475 3      ! Allocate memory for and read in index file bitmap.
373      1476 3      !
374      1477 3      FAST_IMAP[.RVN-1] = GET_VM(.FAST_IMAP_SIZE[.RVN-1] * 512);
375      1478 3      STATUS = $QIOW(
376      1479 3          FUNC=IOS_READVBLK,
377      1480 3          CHAN=.INPUT_CHAN,
378      1481 3          IOSB=IOSB,
379      1482 3          P1=.FAST_IMAP[.RVN-1],
380      1483 3          P2=.FAST_IMAP_SIZE[.RVN-1] * 512,
381      1484 3          P3=.BITMAP_OFFSET);
382      1485 3      IF .STATUS THEN STATUS = .IOSB[0];
383      1486 3      IF NOT .STATUS
384      1487 3      THEN
385      1488 3          SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
386      1489 3
387      1490 3
388      1491 3      ! Read the index file header and get the EOF stored therein.
389      1492 3      !
390      1493 3      STATUS = $QIOW(
391      1494 3          FUNC=IOS_READVBLK,
392      1495 3          CHAN=.INPUT_CHAN,
393      1496 3          IOSB=IOSB,
394      1497 3          P1=.FAST_BUFFER + 512,
395      1498 3          P2=512,
396      1499 3          P3=.FAST_HDR_OFFSET[.RVN-1] + 1);
```



```

397 1500 3 IF .STATUS THEN STATUS = .IOSB[0];
398 1501 3 IF NOT .STATUS
399 1502 3 THEN
400 1503 3     SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
401 1504 3
402 1505 3
403 1506 3 EOF = 0;
404 1507 3 IF .FAST_STRUCLEV EQL 2
405 1508 3     THEN EOF = ROT(.BBLOCK[BBLOCK[.FAST_BUFFER + 512, FH2$W_RECATTR], FAT$L_EFBLK], 16) - 1
406 1509 3     ELSE IF .FAST_BUFFER[HM2$B_STRUCVER] EQL 2
407 1510 3         THEN EOF = ROT(.STATBLK[SBK$L_FILESIZE], 16);
408 1511 3
409 1512 3
410 1513 3 ! Now scan the volume's index file bitmap backwards, looking for the
411 1514 3 ! highest set bit. The maximum of the EOF mark and the highest set
412 1515 3 ! bit is taken to be the true index file EOF.
413 1516 3
414 1517 3 DECR J FROM .FAST_IMAP_SIZE[.RVN-1] * 128 - 1 TO 0 DO
415 1518 4 BEGIN
416 1519 4 IF .VECTOR[.FAST_IMAP[.RVN-1], .J] NEQ 0
417 1520 4 THEN
418 1521 5 BEGIN
419 1522 5 EOF = MAXU(
420 1523 5     .J*32 +
421 1524 5     LEFT_ONE(.VECTOR[.FAST_IMAP[.RVN-1], .J]) +
422 1525 5     .FAST_HDR_OFFSET[.RVN-1],
423 1526 5     .EOF);
424 1527 5 EXITLOOP;
425 1528 5 END;
426 1529 4 END;
427 1530 3
428 1531 3
429 1532 3 IF .QUAL[QUAL_IMAG]
430 1533 3 THEN
431 1534 4 BEGIN
432 1535 4
433 1536 4 ! Read the boot block.
434 1537 4
435 1538 4 STATUS = $QIOW(
436 1539 4     FUNC=IOS$ READVBLK,
437 1540 4     CHAN=.INPUT_CHAN,
438 1541 4     IOSB=IOSB,
439 1542 4     P1=.FAST_BUFFER + 512,
440 1543 4     P2=512,
441 1544 4     P3=1);
442 1545 4 IF .STATUS THEN STATUS = .IOSB[0];
443 1546 4 IF NOT .STATUS
444 1547 4 THEN
445 1548 4     SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
446 1549 4
447 1550 4
448 1551 4 ! If the volume is bootable, save the LBN from the boot block.
449 1552 4
450 1553 4 FAST_BOOT_LBN[.RVN-1] = -1;
451 1554 4 IF CH$RCHAR(.FAST_BUFFER + 512 + 5) EQL 0
452 1555 4 THEN
453 1556 4     FAST_BOOT_LBN[.RVN-1] = ROT(.VECTOR[.FAST_BUFFER + 512, 1], 16);
```

```

454      1557 3      END;
455      1558 3
456      1559 3
457      1560 3      ! Deaccess the index file.
458      1561 3      !
459      P 1562 3      $QIOW(
460      P 1563 3      FUNC=IOS_DEACCESS,
461      1564 3      CHAN=.INPUT_CHAN);
462      1565 3
463      1566 3
464      1567 3      IF .QUAL[QUAL_IMAG]
465      1568 3      THEN
466      1569 4      BEGIN
467      1570 4
468      1571 4      ! Access the bitmap file.
469      1572 4      !
470      1573 4      CH$FILL (0, FIB$C_LENGTH, FIB);
471      1574 4      FIB[FIB$L_ACCTL] = .ACCTL;
472      1575 4      FIB[FIB$W_FID_NUM] = FID$C_BITMAP;
473      1576 4      FIB[FIB$W_FID_SEQ] = FID$C_BITMAP;
474      1577 4      FIB[FIB$W_FID_RVN] = .RVN;
475      P 1578 4      STATUS = $QIOW(
476      P 1579 4      FUNC=IOS_ACCESS OR IOSM_ACCESS,
477      P 1580 4      CHAN=.INPUT_CHAN,
478      P 1581 4      IOSB=IOSB,
479      1582 4      P1=FIB_DESC);
480      1583 4      IF .STATUS THEN STATUS = .IOSB[0];
481      1584 4      IF NOT .STATUS
482      1585 4      THEN
483      1586 4      SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
484      1587 4
485      1588 4
486      1589 4      ! Read the storage control block.
487      1590 4      !
488      P 1591 4      STATUS = $QIOW(
489      P 1592 4      FUNC=IOS_READVBLK,
490      P 1593 4      CHAN=.INPUT_CHAN,
491      P 1594 4      IOSB=IOSB,
492      P 1595 4      P1=.FAST_BUFFER + 1024,
493      P 1596 4      P2=512,
494      1597 4      P3=1);
495      1598 4      IF .STATUS THEN STATUS = .IOSB[0];
496      1599 4      IF NOT .STATUS
497      1600 4      THEN
498      1601 4      SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
499      1602 4
500      1603 4
501      1604 4      ! Deaccess the bitmap file.
502      1605 4      !
503      P 1606 4      $QIOW(
504      P 1607 4      FUNC=IOS_DEACCESS,
505      1608 4      CHAN=.INPUT_CHAN);
506      1609 4
507      1610 4
508      1611 4      ! Generate the volume attribute record.
509      1612 4      !
510      1613 4      IF NOT .QUAL[QUAL_VOLU] OR .QUAL[QUAL_VOLU_VALUE] EQL .RVN
```



```

511      1614 4      THEN
512      1615 4          VOLUME_ATTRIBUTES(
513      1616 4              .FAST_BUFFER,      ! home block
514      1617 4              .FAST_BUFFER+512,    ! boot block
515      1618 4              .FAST_BUFFER+1024,    ! storage control block
516      1619 4              .EOF = .FAST_HDR_OFFSET[.RVN-1]);
517      1620 4      END;
518      1621 4
519      1622 4
520      1623 4      ! Access the index file.
521      1624 4      !
522      1625 4      CH$FILL (0, FIB$C_LENGTH, FIB);
523      1626 4      FIB[FIB$L_ACCTL] = .ACCTL;
524      1627 4      FIB[FIB$W_FID_NUM] = FID$C_INDEXF;
525      1628 4      FIB[FIB$W_FID_SEQ] = FID$C_INDEXF;
526      1629 4      FIB[FIB$W_FID_RVN] = .RVN;
527      1630 4      STATUS = $QIOW(
P      1631 4          FUNC=IOS_ACCESS OR IOSM_ACCESS,
P      1632 4          CHAN=.INPUT_CHAN,
P      1633 4          IOSB=IOSB,
531      1634 4          P1=FIB_DESC);
532      1635 4      IF .STATUS THEN STATUS = .IOSB[0];
533      1636 4      IF NOT .STATUS
534      1637 4      THEN
535      1638 4          SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
536      1639 4
537      1640 4
538      1641 4      ! Loop for all headers in the index file. Read multiple blocks into a
539      1642 4      ! buffer and process them one at a time.
540      1643 4      !
541      1644 4      DIR_STATUS = 0;      ! Used by SELECT_INPUT_FILE
542      1645 4      VBN = .FAST_HDR_OFFSET[.RVN-1] + 1;
543      1646 4      UNTIL .VBN GTRU .EOF DO
544      1647 4          BEGIN
545      1648 4              LOCAL
546      1649 4                  HEADER:      REF BBLOCK,      ! Pointer to header
547      1650 4                  READ_COUNT;      ! Blocks to read on current iteration
548      1651 4
549      1652 4
550      1653 4      ! Establish the count of blocks to read and execute the read.
551      1654 4      !
552      1655 4      READ_COUNT = MINU(INDEX_BUF_COUNT, .EOF + 1 - .VBN);
553      1656 4      STATUS = $QIOW(
P      1657 4          FUNC=IOS_READVBLK,
P      1658 4          CHAN=.INPUT_CHAN,
P      1659 4          IOSB=IOSB,
557      1660 4          P1=.FAST_BUFFER,
558      1661 4          P2=512 * .READ_COUNT,
559      1662 4          P3=.VBN);
560      1663 4      IF .STATUS THEN STATUS = .IOSB[0];
561      1664 4
562      1665 4
563      1666 4      ! If an error occurred, read each block separately, reporting any
564      1667 4      ! errors.
565      1668 4      !
566      1669 4      IF NOT .STATUS
567      1670 4      THEN
```



```
568      BEGIN
569      INCR XVBN FROM 0 TO .READ_COUNT-1 DO
570      BEGIN
571      LOCAL
572      HEADER:      REF BBLOCK;      ! Pointer to header
573
574      HEADER = .FAST_BUFFER + .XVBN * 512;
575      STATUS = $QIOWT
576      P  FUNC=IOS_READVBLK,
577      P  CHAN=.INPUT_CHAN,
578      P  IOSB=IOSB,
579      P  P1=.HEADER,
580      P  P2=512,
581      P  P3=.VBN + .XVBN);
582      IF .STATUS THEN STATUS = .IOSB[0];
583      IF NOT .STATUS
584      THEN
585      CH$FILL(0, 512, .HEADER);
586      END;
587      END;
588
589      ! For each header, verify that it is a valid file header.
590      ! If it is, process it.
591      HEADER = .FAST_BUFFER;
592      INCR XVBN FROM 0 TO .READ_COUNT-1 DO
593      BEGIN
594      LOCAL
595      MAP_AREA:      REF BBLOCK,      ! Pointer to map area
596      FILE_NUMBER,      ! Current file number
597      FILE_ID:      BBLOCK[FID$C_LENGTH];      ! Current file ID
598
599      ! Get a clean file ID.
600      FILE_NUMBER = .VBN + .XVBN - .FAST_HDR_OFFSET[.RVN-1];
601      FILE_ID[FID$W_NUM] = .FILE_NUMBER<0,16>;
602      FILE_ID[FID$B_NMX] = .FILE_NUMBER<16,8>;
603      IF .FAST_STRUCLEV EQL 2
604      THEN FILE_ID[FID$W_SEQ] = .HEADER[FH2$W_FID_SEQ]
605      ELSE FILE_ID[FID$W_SEQ] = .HEADER[FH1$W_FID_SEQ];
606      FILE_ID[FID$B_RVN] = .RVN;
607
608      ! Generate the FID record for an image mode save.
609      IF
610      .QUAL[QUAL_IMAG] AND
611      .XVBN EQL 0 AND
612      (NOT .QUAL[QUAL_VOLU] OR .QUAL[QUAL_VOLU_VALUE] EQL .RVN)
613      THEN
614      GEN_FID_RECORD(.HEADER, .READ_COUNT, .FILE_NUMBER, .RVN);
615
616      ! Validate the header. In ODS-2, a valid header is to be taken as
```



```

: 625      1728 5      ! valid even if the index file bitmap shows it as available. In
: 626      1729 5      ! ODS-1, a header corresponding to a clear index file bitmap bit
: 627      1730 5      ! is not to be examined.
: 628      1731 5
: 629      1732 5      STATUS = VERIFY_HEADER(.HEADER, FILE_ID);
: 630      1733 5      IF
: 631      1734 5          .FAST_STRUCLEV EQL 1 AND
: 632      1735 5          NOT .BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1]
: 633      1736 5      THEN
: 634      1737 5          STATUS = FALSE;
: 635      1738 5
: 636      1739 5      ! Clear the index file bitmap bit until we determine that the
: 637      1740 5      ! header is not an extension header.
: 638      1741 5
: 639      1742 5      BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1] = FALSE;
: 640      1743 5
: 641      1744 5
: 642      1745 5
: 643      1746 5      IF .STATUS
: 644      1747 5      THEN
: 645      1748 6          BEGIN
: 646      1749 6              ! Header is valid.
: 647      1750 6              ! Other processing executed only if not an extension header.
: 648      1751 6
: 649      1752 6              MAP_AREA = .HEADER + .HEADER[FH1$B_MPOFFSET]*2;
: 650      1753 6              IF
: 651      1754 6                  BEGIN
: 652      1755 7                  IF .FAST_STRUCLEV EQL 2
: 653      1756 7                      THEN .HEADER[FH2$W_SEG_NUM] EQL 0
: 654      1757 7                      ELSE .MAP_AREA[FH1$B_EX_SEGNUM] EQL 0
: 655      1758 7                  END
: 656      1759 7              THEN
: 657      1760 6                  BEGIN
: 658      1761 7                      LOCAL
: 659      1762 7                          FCH:      REF BBLOCK;      ! Pointer to characteristics
: 660      1763 7
: 661      1764 7
: 662      1765 7
: 663      1766 7              ! Make sure the index file bitmap indicates that the header
: 664      1767 7              ! is valid.
: 665      1768 7
: 666      1769 7              BITVECTOR[FAST_IMAP[RVN-1], .FILE_NUMBER-1] = TRUE;
: 667      1770 7
: 668      1771 7
: 669      1772 7              ! Get file characteristics pointer.
: 670      1773 7
: 671      1774 7              IF .FAST_STRUCLEV EQL 2
: 672      1775 7                  THEN FCH = HEADER[FH2$L_FILECHAR]
: 673      1776 7                  ELSE FCH = HEADER[FH1$W_FILECHAR];
: 674      1777 7
: 675      1778 7
: 676      1779 7              ! Evaluate file selection criteria for header. However,
: 677      1780 7              ! always reject files marked for delete. If this is /IMAGE
: 678      1781 7              ! mode, select all other valid files.
: 679      1782 7
: 680      1783 7              IF
: 681      1784 8                  BEGIN
```



```

: 682      1785      8      IF .FCH[FCH$V_MARKDEL]
: 683      1786      8      THEN
: 684      1787      8          TRUE
: 685      1788      8      ELSE
: 686      1789      9          BEGIN
: 687      1790      9              INIT_ATTR(.HEADER);
: 688      1791      9              IF .QUAL[QUAL_IMAG]
: 689      1792      9                  THEN FALSE
: 690      1793      9                  ELSE NOT SELECT_INPUT_FILE(%B'001')
: 691      1794      9          END
: 692      1795      8      END
: 693      1796      7      THEN
: 694      1797      7          BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] = FALSE;
: 695      1798      6      END;
: 696      1799      5      END;
: 697      1800      5
: 698      1801      5
: 699      1802      5          HEADER = .HEADER + 512;
: 700      1803      4          END;
: 701      1804      4
: 702      1805      4
: 703      1806      4          VBN = .VBN + INDEX_BUF_COUNT;
: 704      1807      3          END;
: 705      1808      3
: 706      1809      3
: 707      1810      3      ! Deaccess the index file.
: 708      1811      3      !
: 709      1812      3      $QIOW(
P 710      1813      3          FUNC=IOS$ DEACCESS,
P 711      1814      3          CHAN=.INPUT_CHAN);
: 712      1815      3
: 713      1816      3
: 714      1817      3          RVN = .RVN + 1;
: 715      1818      3      END
: 716      1819      2      WHILE .RVN LEQU .COM_I_SETCOUNT;
: 717      1820      2
: 718      1821      2
: 719      1822      2      ! Free the general buffer.
: 720      1823      2      !
: 721      1824      2      FREE_VM(.FAST_BUFFER_SIZE, .FAST_BUFFER);
: 722      1825      2      FAST_BUFFER = FAST_BUFFER_SIZE = 0;
: 723      1826      2      COM_FLAGS[COM_DSBL_CHKPT] = FALSE;
: 724      1827      2
: 725      1828      2
: 726      1829      2      ! Scan all directories on all volumes.
: 727      1830      2      !
: 728      1831      2      INCR RVN FROM 1 TO .COM_I_SETCOUNT DO DIR_SCAN(.RVN);
: 729      1832      2
: 730      1833      2
: 731      1834      2      ! Allocate a buffer to read file headers.
: 732      1835      2      !
: 733      1836      2      FAST_BUFFER_SIZE = 512;
: 734      1837      2      FAST_BUFFER = GET_VM(512);
: 735      1838      2
: 736      1839      2
: 737      1840      2      ! If this is /IMAGE mode, scan for and process lost files.
: 738      1841      2      !

```

```

: 739      1842  2 IF .QUAL[QUAL_IMAG]
: 740      1843  2 THEN
: 741      1844  3 BEGIN
: 742      1845  3 INCR RVN FROM 1 TO .COM_I_SETCOUNT DO
: 743      1846  4 BEGIN
: 744      1847  4
: 745      1848  4 ! Scan bitmap for files not yet processed and process these.
: 746      1849  4
: 747      1850  4 INCR FILE_NUMBER FROM 1 TO .FAST_IMAP_SIZE[.RVN-1]*4096 DO
: 748      1851  5 BEGIN
: 749      1852  5 IF .BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1]
: 750      1853  5 THEN
: 751      1854  6 BEGIN
: 752      1855  6 LOCAL
: 753      1856  6 NAME_LENGTH, ! Length of filename
: 754      1857  6 NAME_ADDRESS, ! Address of filename
: 755      1858  6 RSA_DESC: VECTOR[2], ! Descriptor for RSA
: 756      1859  6 FILENAME: VECTOR[20, BYTE]; ! Filename buffer
: 757      1860  6
: 758      1861  6
: 759      1862  6 ! Access index file on current RVN.
: 760      1863  6
: 761      1864  6 CH$FILL (0, FIB$C_LENGTH, FIB);
: 762      1865  6 FIB[FIB$L_ACCTL] = FIB$M_NORECORD;
: 763      1866  6 FIB[FIB$W_FID_NUM] = FIB$C_INDEXF;
: 764      1867  6 FIB[FIB$W_FID_SEQ] = FIB$C_INDEXF;
: 765      1868  6 FIB[FIB$W_FID_RVN] = .RVN;
: 766      1869  6 STATUS = $QIOW(
: 767      1870  6 FUNC=IOS_ACCESS OR IOSM_ACCESS,
: 768      1871  6 CHAN=.INPUT_CHAN,
: 769      1872  6 IOSB=IOSB,
: 770      1873  6 P1=FIB_DESC);
: 771      1874  6 IF .STATUS THEN STATUS = .IOSB[0];
: 772      1875  6 IF NOT .STATUS
: 773      1876  6 THEN
: 774      1877  6 SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 775      1878  6
: 776      1879  6
: 777      1880  6 ! Read file header.
: 778      1881  6
: 779      1882  6 STATUS = $QIOW(
: 780      1883  6 FUNC=IOS_READVBLK,
: 781      1884  6 CHAN=.INPUT_CHAN,
: 782      1885  6 IOSB=IOSB,
: 783      1886  6 P1=.FAST_BUFFER,
: 784      1887  6 P2=512,
: 785      1888  6 P3=.FAST_HDR_OFFSET[.RVN-1] + .FILE_NUMBER);
: 786      1889  6 IF .STATUS THEN STATUS = .IOSB[0];
: 787      1890  6 IF NOT .STATUS
: 788      1891  6 THEN
: 789      1892  6 SIGNAL(BACKUP$_PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .STATUS);
: 790      1893  6
: 791      1894  6
: 792      1895  6 ! Deaccess index file on current RVN.
: 793      1896  6
: 794      1897  6 $QIOW(
: 795      1898  6 FUNC=IOS_DEACCESS,
```



```
.. 796      1899  6      CHAN=.INPUT_CHAN);
.. 797      1900  6
.. 798      1901  6
.. 799      1902  6      ! Set up file ID.
.. 800      1903  6      !
.. 801      1904  6      INPUT_NAM[NAM$W_FID_NUM] = .FILE_NUMBER;
.. 802      1905  6      INPUT_NAM[NAM$B_FID_NMX] = .FILE_NUMBER<16,8>;
.. 803      1906  6      INPUT_NAM[NAM$B_FID_RVN] = .RVN;
.. 804      1907  6      INPUT_NAM[NAM$W_DID_NUM] = 0;
.. 805      1908  6      INPUT_NAM[NAM$W_DID_SEQ] = 0;
.. 806      1909  6      INPUT_NAM[NAM$W_DID_RVN] = 0;
.. 807      1910  6
.. 808      1911  6
.. 809      1912  6      ! Generate filename string from file header ident area.
.. 810      1913  6      ! Also get file sequence number.
.. 811      1914  6      !
.. 812      1915  6      IF .FAST_STRUCLEV EQL 2
.. 813      1916  6      THEN
.. 814      1917  7      BEGIN
.. 815      1918  7      LOCAL
.. 816      1919  7      P;
.. 817      1920  7
.. 818      1921  7      INPUT_NAM[NAM$W_FID_SEQ] = .FAST_BUFFER[FH2$W_FID_SEQ];
.. 819      1922  7      NAME_ADDRESS =
.. 820      1923  7      BBLOCK[.FAST_BUFFER + .FAST_BUFFER[FH2$B_IDOFFSET]*2,
.. 821      1924  7      F12$T_FILENAME];
.. 822      1925  7      NAME_LENGTH = F12$S_FILENAME;
.. 823      1926  7      P = CH$FIND_CH(F12$S_FILENAME, .NAME_ADDRESS, %C' ');
.. 824      1927  7      IF .P NEQ 0 THEN NAME_LENGTH = .P - .NAME_ADDRESS;
.. 825      1928  7      END
.. 826      1929  6      ELSE
.. 827      1930  7      BEGIN
.. 828      1931  7      INPUT_NAM[NAM$W_FID_SEQ] = .FAST_BUFFER[FH1$W_FID_SEQ];
.. 829      1932  7      NAME_ADDRESS = FILENAME;
.. 830      1933  7      NAME_LENGTH = MAKE_STRING(
.. 831      1934  7      BBLOCK[.FAST_BUFFER + .FAST_BUFFER[FH1$B_IDOFFSET]*2,
.. 832      1935  7      F11$W_FILENAME] - $BYTEOFFSET(NMB$W_NAME),
.. 833      1936  7      FILENAME) AND 65535;
.. 834      1937  6      END;
.. 835      1938  6      RSA_DESC[0] = NAM$C_MAXRSS;
.. 836      1939  6      RSA_DESC[1] = .INPUT_NAM[NAM$L_RSA];
.. 837      1940  6      $FAD(
.. 838      1941  6      $DESCRIPTOR('!ASC[!AD'),
.. 839      1942  6      RSA_DESC,
.. 840      1943  6      RSA_DESC,
.. 841      1944  6      INPUT_QUAL[QUAL_DEV_DESC],
.. 842      1945  6      .NAME_LENGTH, .NAME_ADDRESS);
.. 843      1946  6      INPUT_NAM[NAM$B_RSL] = .RSA_DESC[0];
.. 844      1947  6      INIT_NAMEBLOCK(.INPUT_NAM);
.. 845      1948  6
.. 846      1949  6
.. 847      1950  6      ! Call the routine to process the file.
.. 848      1951  6      !
.. 849      1952  6      SAVE_ONE_FILE();
.. 850      1953  5      END;
.. 851      1954  4      END;
.. 852      1955  3      END;
```

```

: 853      1956 2      END;
: 854      1957 2
: 855      1958 2
: 856      1959 2 ! Free the file header buffer.
: 857      1960 2
: 858      1961 2 FREE_VM(.FAST_BUFFER_SIZE, .FAST_BUFFER);
: 859      1962 2 FAST_BUFFER = FAST_BUFFER_SIZE = 0;
: 860      1963 2
: 861      1964 2
: 862      1965 2 ! Free memory for index file bitmaps.
: 863      1966 2
: 864      1967 2 INCR RVN FROM 1 TO .COM_I_SETCOUNT DO
: 865      1968 2 BEGIN
: 866      1969 2 FREE_VM(.FAST_IMAP_SIZE[RVN-1] * 512, .FAST_IMAP[RVN-1]);
: 867      1970 2 FAST_IMAP[RVN-1] = 0;
: 868      1971 2 END;
: 869      1972 2
: 870      1973 2
: 871      1974 2 ! Free the per-volume data.
: 872      1975 2
: 873      1976 2 INCRA A FROM FAST_VOL_BEG TO FAST_VOL_END-%UPVAL BY %UPVAL DO
: 874      1977 2 BEGIN
: 875      1978 2 FREE_VM(.COM_I_SETCOUNT*%UPVAL, ..A);
: 876      1979 2 A = 0;
: 877      1980 2 END;
: 878      1981 1 END;
```

```

.TITLE FASTSCAN Fast file scan
.IDENT \V04-000\
.PSECT COMMON,NOEXE, OVR,2
```

```

00000 GLOBAL_BASE:
      .BLKB 0
00000 FREE_LIST:
      .BLKB 8
00008 INPUT_WAIT:
      .BLKB 8
00010 REREAD_WAIT:
      .BLKB 8
00018 OUTPUT_WAIT:
      .BLKB 8
00020 JPI_UIC:
      .BLKB 4
00024 JPI_USERNAME:
      .BLKB 12
00030 JPI_DATE:
      .BLKB 8
00038 JPI_NODE_DESC:
      .BLKB 8
00040 JPI_CURPRIV:
      .BLKB 8
00048 SYI_VERSION:
      .BLKB 4
0004C SYI_SID:
      .BLKB 4
00050 RWSV_HOLD_LIST:
      .BLKB 8
```



00058 RWSV\_CRC16:  
          .BLKB 64  
00098 RWSV\_AUTODIN:  
          .BLKB 64  
000D8 RWSV\_FILESET\_ID:  
          .BLKB 8  
000E0 RWSV\_VOLUME\_ID:  
          .BLKB 12  
000EC RWSV\_VOL\_NUMBER:  
          .BLKB 2  
000EE RWSV\_SEG\_NUMBER:  
          .BLKB 2  
000F0 RWSV\_FILE\_NUMBER:  
          .BLKB 4  
000F4 RWSV\_SAVE\_QUAL:  
          .BLKB 4  
000F8 RWSV\_SAVE\_FAB:  
          .BLKB 4  
000FC RWSV\_CHAN:  
          .BLKB 4  
00100 RWSV\_XOR\_BCB:  
          .BLKB 4  
00104 RWSV\_IN\_SEQ:  
          .BLKB 4  
00108 RWSV\_IN\_SEQ 0:  
          .BLKB 4  
0010C RWSV\_IN\_XOR\_SEQ:  
          .BLKB 4  
00110 RWSV\_IN\_XOR\_RFA:  
          .BLKB 6  
00116 RWSV\_LOOKAHEAD:  
          .BLKB 1  
00117 RWSV\_XOR\_SIZE:  
          .BLKB 1  
00118 RWSV\_IN\_GROUP\_SIZE:  
          .BLKB 4  
0011C RWSV\_IN\_ERRORS:  
          .BLKB 2  
0011E RWSV\_IN\_XORUSE:  
          .BLKB 2  
00120 RWSV\_IN\_ORGERR:  
          .BLKB 8  
00128 RWSV\_IN\_VBN:  
          .BLKB 4  
0012C RWSV\_IN\_VBN 0:  
          .BLKB 4  
00130 RWSV\_ALLOC:  
          .BLKB 4  
00134 RWSV\_EOF:  
          .BLKB 4  
00138 RWSV\_OUT\_SEQ:  
          .BLKB 4  
0013C RWSV\_OUT\_VBN:  
          .BLKB 4  
00140 RWSV\_OUT\_BLOCK\_COUNT:  
          .BLKB 4  
00144 RWSV\_OUT\_ERRORS:



```
00146 RWSV_SEQ_ERRORS: 2
00148 RWSV_OUT_GROUP_COUNT: 2
00149 RWSV_PADDING: 1
0014C QUAL: 3
001BC COM_SSNAME: 112
001C4 COM_VALID_TYPES: 8
001C6 COM_FLAGS: 2
001C8 COM_PADDING: 2
001C9 COM_BUFF_COUNT: 1
001CA COM_I_SETCOUNT: 1
001CB COM_O_SETCOUNT: 1
001CC COM_I_STRUCNAME: 1
001D8 COM_O_STRUCNAME: 12
001E4 COM_O_BSRDATE: 12
001EC ALT_SSNAME: 8
0020C INPUT_FUNC: 32
0020D INPUT_RTYPE: 1
0020E OUTPUT_FUNC: 1
0020F FAST_STRUCLEV: 1
00210 INPUT_BEG: 1
00210 INPUT_CHAN: 0
00214 INPUT_FLAGS: 4
00216 INPUT_PADDING: 2
00218 INPUT_FAB: 2
0021C INPUT_NAM: 4
00220 INPUT_BCB: 4
00224 INPUT_QUAL: 4
00228 INPUT_BAD: 4
0022C INPUT_BLOCK: 4
```



00230	INPUT_MAXBLOCK:	.BLKB	4
00234	INPUT_MEDIA_ID:	.BLKB	4
00238	INPUT_NAMEDESC:	.BLKB	8
00240	INPUT_STATBLK:	.BLKB	8
00248	INPUT_HDR_BEG:	.BLKB	0
00248	INPUT_CRDATE:	.BLKB	8
00250	INPUT_REVDATE:	.BLKB	8
00258	INPUT_EXPDATE:	.BLKB	8
00260	INPUT_BAKDATE:	.BLKB	8
00268	INPUT_FILEOWNER:	.BLKB	4
0026C	INPUT_FILECHAR:	.BLKB	4
00270	INPUT_RECATTR:	.BLKB	32
00290	INPUT_HDR_END:	.BLKB	0
00290	INPUT_END:	.BLKB	0
00290	INPUT_PROG_LIST:	.BLKB	4
00294	INPUT_PLACEMENT:	.BLKB	8
0029C	INPUT_VBN_LIST:	.BLKB	8
002A4	INPUT_PLACE_LEN:	.BLKB	2
002A6	INPUT_PADDING_2:	.BLKB	2
002A8	OUTPUT_BEG:	.BLKB	0
002A8	OUTPUT_CHAN:	.BLKB	4
002AC	OUTPUT_FLAGS:	.BLKB	2
002AE	OUTPUT_PADDING:	.BLKB	2
002B0	OUTPUT_FAB:	.BLKB	4
002B4	OUTPUT_NAM:	.BLKB	4
002B8	OUTPUT_BCB:	.BLKB	4
002BC	OUTPUT_QUAL:	.BLKB	4
002C0	OUTPUT_BAD:	.BLKB	4

002C4 OUTPUT\_BLOCK:  
          .BKLB 4  
002C8 OUTPUT\_MAXBLOCK:  
          .BKLB 4  
002CC OUTPUT\_DEVGEOM:  
          .BKLB 8  
002D4 OUTPUT\_ATTBUF:  
          .BKLB 144  
00364 OUTPUT\_END:  
          .BKLB 0  
00364 LIST\_TOTFILES:  
          .BKLB 4  
00368 LIST\_TOTSIZE:  
          .BKLB 4  
0036C VERIFY\_FAB:  
          .BKLB 4  
00370 VERIFY\_USE\_COUNT:  
          .BKLB 4  
00374 VERIFY\_QUAL:  
          .BKLB 4  
00378 COMPARE\_BCB:  
          .BKLB 4  
0037C FAST\_BUFFER:  
          .BKLB 4  
00380 FAST\_BUFFER\_SIZE:  
          .BKLB 4  
00384 FAST\_RVN:  
          .BKLB 1  
00385 FAST\_PADDING:  
          .BKLB 1  
00386 DIR\_VERLIMIT:  
          .BKLB 2  
00388 FAST\_VOL\_BEG:  
          .BKLB 0  
00388 FAST\_IMAP\_SIZE:  
          .BKLB 4  
0038C FAST\_IMAP:  
          .BKLB 4  
00390 FAST\_HDR\_OFFSET:  
          .BKLB 4  
00394 FAST\_BOOT\_LBN:  
          .BKLB 4  
00398 FAST\_VOL\_END:  
          .BKLB 0  
00398 JOUR\_BUFFER:  
          .BKLB 4  
0039C JOUR\_DIR:  
          .BKLB 4  
003A0 JOUR\_HIBLK:  
          .BKLB 4  
003A4 JOUR\_EFBLK:  
          .BKLB 4  
003A8 JOUR\_INBLK:  
          .BKLB 4  
003AC JOUR\_FFBYTE:  
          .BKLB 2  
003AE JOUR\_INBYTE:



003B0	JOUR_STRUCT_LEV:	.BLKB	2
003B2	JOUR_COUNT:	.BLKB	2
003B3	JOUR_REVERSE:	.BLKB	1
003B4	JOUR_EXSZ:	.BLKB	1
003B6	JOUR_PADDING:	.BLKB	2
003B8	CHKPT_HIGH_SP:	.BLKB	2
003BC	CHKPT_LOW_SP:	.BLKB	4
003C0	CHKPT_STACK:	.BLKB	4
003C4	CHKPT_VARS:	.BLKB	4
003C8	CHKPT_STATUS:	.BLKB	4
003CC	DIR_BEG:	.BLKB	0
003CC	DIR_CHAN:	.BLKB	4
003D0	DIR_NAM:	.BLKB	4
003D4	DIR_DEV_DESC:	.BLKB	4
003D8	DIR_SEL_DIR:	.BLKB	8
003E0	DIR_SEL_NTV:	.BLKB	8
003E8	DIR_STRUCLEV:	.BLKB	1
003E9	DIR_LEVELS:	.BLKB	1
003EA	DIR_FLAGS:	.BLKB	1
003EB	DIR_STATUS:	.BLKB	1
003EC	DIR_STRING:	.BLKB	320
0052C	DIR_STACK:	.BLKB	612
00790	DIR_SP:	.BLKB	4
00794	DIR_SEL_LATEST:	.BLKB	4
00798	DIR_END:	.BLKB	0
00798	DIR_SCANLIMIT:	.BLKB	36
007BC	INPUT_MTL:	.BLKB	4
007C0	OUTPUT_MTL:	.BLKB	4
007C4	CURRENT_MTL:	.BLKB	4
007C8	CURRENT_VCB:	.BLKB	4

```
007CC CURRENT_WCB:
007D0 ACL_FIB_DESCR:
007D8 ACL_FIB:
00818 ACL_LENGTH:
0081C ACL_BUFFER:
00820 CRYPT_IN_CONTEXT:
00824 CRYPT_OUT_CONTEXT:
00828 CRYPT_DATA_CONTEXT:
0082C CRYPT_DATA_ENCIV:
00834 CRYPT_DATA_CODE:
00838 CRYPT_DATA_KEY:
00840 CRYPT_DATA_IV:
00848 CRYPT_DATA_CKSM:

.PSECT CODE,NOWRT,2

44 30 30 30 30 30 30 06 00000 P.AAA: .ASCII <6>\0000000\
44 41 21 5D 5B 53 41 21 00007 P.AAC: .ASCII \!ASC\!AD\
0000F .BLKB 1
00000008 00010 P.AAB: .LONG 8
00000000 00014 .ADDRESS P.AAC

MFD= P.AAA
.EXTRN LIB$EXTRACT CONCEALED
.EXTRN CHECKSUM, CHECKSUM2
.EXTRN GEN_FID_RECORD, INIT_NAMEBLOCK
.EXTRN VOLUME_ATTRIBUTES
.EXTRN SAVE_ONE_FILE, FILE_ERROR
.EXTRN INIT_ATTR, LEFT_ONE
.EXTRN MAKE_STRING, INIT_SEL_INFO
.EXTRN MATCH_DIRECTORY
.EXTRN MATCH_FILENAME, SELECT_INPUT_FILE
.EXTRN TERMINATE_SCAN, FREE_VM
.EXTRN GET_VM, GET_ZERO_VM
.EXTRN LIB$SIGNAL, BACKUP$MAXVOLS
.EXTRN BACKUP$PROCINDEX
.EXTRN BACKUP$OPENDIR
.EXTRN BACKUP$OPENIN, BACKUP$READDIR
.EXTRN BACKUP$BADDIR, BACKUP$NOSUCHRVN
.EXTRN LIB$INVFILSPE, SYSSQIOQ
.EXTRN SYSSFAO

OFFC 00000 .ENTRY FAST_FILE_SCAN, Save R2,R3,R4,R5,R6,R7,R8,- : 1330
SE 90 AE 9E 00002 MOVAB R9,R10,R11
-112(SP), SP
```



0040	8F	00	28 AE 40 8F 9A 00006	MOVZBL #64, FIB_DESC	1369
			2C AE 30 AE 9E 0000B	MOVAB FIB, FIB_DESC+4	1370
			00000000' EF 40 8F 88 00010	BISB2 #64, COM_FLAGS	1375
			00000000' EF 8000 8F 3C 00018	MOVZWL #32768, FAST_BUFFER_SIZE	1376
			00000000G 7E 8000 8F 3C 00021	MOVZWL #32768, -(SP)	1377
			00000000' 00 EF 01 FB 00026	CALLS #1, GET_VM	
			00000000' 58 EF 50 D0 0002D	MOVL R0, FAST_BUFFER	
			6E 58 01 D0 00034	MOVL #1, RVN	1382
			30 AE 00 2C 00037 1\$:	MOVCS #0, (SP), #0, #64, FIB	1397
			56 00200100 8F D0 00040	MOVL #2097408, ACCTL	1398
			30 AE 56 D0 00047	MOVL ACCTL, FIB	1399
			34 AE 00010001 8F D0 0004B	MOVL #65537, FIB+4	1400
			38 AE 58 B0 00053	MOVW RVN, FIB+8	1402
			14 AE 00090008 8F D0 00057	MOVL #589832, ATR_DESC	1403
			18 AE 0C AE 9E 0005F	MOVAB STATBLK, ATR_DESC+4	1405
			1C AE D4 00064	CLRL ATR_DESC+8	1406
			18 7E D4 00067	CLRL -(SP)	1412
			7E 9F 00069	PUSHAB ATR_DESC	
			7E 7C 0006C	CLRL -(SP)	
			3C 7E D4 0006E	CLRL -(SP)	
			40 AE 9F 00070	PUSHAB FIB_DESC	
			7E 7E 7C 00073	CLRL -(SP)	
			72 8F 9A 00075	PUSHAB IOSB	
			00000000' EF DD 0007C	MOVZBL #114, -(SP)	
			00000000G 00 0C FB 00082	PUSHL INPUT_CHAN	
			5A 50 D0 00084	CLRL -(SP)	
			04 5A E9 0008B	CALLS #12, SYSSQIOW	
			5A 20 AE 3C 00091	MOVL R0, STATUS	1413
			0000025C 8F 5A D1 00095 2\$:	BLBC STATUS, 2\$	1414
			6E 52 12 0009C	MOVZWL IOSB, STATUS	1417
			30 AE 00 2C 0009E	CMPL STATUS, #604	
			56 00200000 8F D0 000A5	BNEQ 3\$	
			30 AE 00200000 8F D0 000A7	MOVCS #0, (SP), #0, #64, FIB	1418
			34 AE 00010001 8F D0 000AE	MOVL #2097152, ACCTL	1419
			38 AE 58 B0 000BE	MOVL #2097152, FIB	1420
			18 7E D4 000C2	MOVL #65537, FIB+4	1422
			3C AE 9F 000C4	MOVW RVN, FIB+8	1428
			40 7E 7C 000C7	CLRL -(SP)	
			72 7E D4 000C9	PUSHAB ATR_DESC	
			00000000G 00 0C FB 000CB	CLRL -(SP)	
			5A 50 D0 000CE	PUSHAB FIB_DESC	
			07 5A E9 000D0	CLRL -(SP)	
			18 5A 3C 000D3	PUSHAB IOSB	
			0500 8F 9A 000D7	MOVZBL #114, -(SP)	
			7E 00000000' EF DD 000DD	PUSHL INPUT_CHAN	
			00000000G 00 0C FB 000DF	CLRL -(SP)	
			5A 50 D0 000E6	CALLS #12, SYSSQIOW	
			07 5A E9 000E9	MOVL R0, STATUS	1429
			18 5A 3C 000EC	BLBC STATUS, 4\$	
			3\$:	MOVZWL IOSB, STATUS	
			4\$:	BLBS STATUS, 5\$	1431
			0500 8F BB 000F3	MOVZWL STATUS, 5\$	1433
			7E 00000000' EF 10 C1 000F7	PUSHR #*M<R8,R10>	
			02 DD 000FF	ADDL3 #16, INPUT_QUAL, -(SP)	
				PUSHL #2	

00000000G	00	00000000G	8F	DD	00101	PUSHL	#BACKUP\$ PROCINDEX		
			05	FB	00107	CALLS	#5, LIB\$SIGNAL		
		00000000'	EF	DD	0010E	PUSHL	FAST_BUFFER	1439	
			58	DD	00114	PUSHL	RVN		
0000V	CF		02	FB	00116	CALLS	#2, READ_HOMEBLOCK		
	01		58	D1	0011B	CMPL	RVN, #1	1446	
			2A	12	0011E	BNEQ	8\$		
		00000000'	EF	9E	00120	MOVAB	FAST_VOL_BEG, R2	1449	
		00000000'	EF	9E	00127	MOVAB	FAST_VOL_END-4, R3		
			15	11	0012E	BRB	7\$		
7E		00000000'	EF	9A	00130	MOVZBL	COM_I SETCOUNT, R0	1451	
			02	78	00137	ASHL	#2, R0, -(SP)		
00000000G	00		01	FB	0013B	CALLS	#1, GET_ZERO_VM		
			50	D0	00142	MOVL	R0, (A)7		
			53	52	D1	00145	7\$:	1449	
			E6	1B	00148	CMPL	A, R3		
		00000000'	EF	D0	0014A	BLEQU	6\$		
		00000000'FF	48	DE	00151	MOVL	FAST_BUFFER, R4	1461	
		00000000'FF	48	DE	00159	MOVAL	@FAST_IMAP_SIZE[RVN], R3	1462	
		00000000'	EF	91	00161	MOVAL	@FAST_HDR_OFFSET[RVN], R1	1464	
			19	12	00168	CMPB	FAST_STRUCLEV, #2	1458	
			A4	3C	0016A	BNEQ	9\$		
		0E	A4	3C	0016E	MOVZWL	14(R4), CLUSTER	1461	
		20	A4	3C	00172	MOVZWL	32(R4), R5	1462	
52	FC		55	D0	00172	MOVL	R5, -4(R3)		
			02	78	00176	ASHL	#2, CLUSTER, BITMAP_OFFSET	1463	
			52	D6	0017A	INCL	BITMAP_OFFSET		
	FC	A1	65	40	DE	0017C	MOVAL	(R5)[CLUSTER], -4(R1)	1464
			12	11	00181	BRB	10\$	1458	
			01	D0	00183	9\$:	MOVL	#1, CLUSTER	1468
	FC	A3	64	3C	00186	MOVZWL	(R4), -4(R3)	1469	
			03	D0	0018A	MOVL	#3, BITMAP_OFFSET	1470	
	FC	A1	64	3C	0018D	MOVZWL	(R4), -4(RT)	1471	
	FC	A1	02	C0	00191	ADDL2	#2, -4(R1)		
		00000000'FF	48	DE	00195	10\$:	MOVAL	@FAST_IMAP[RVN], R4	1477
7E	FC	A3	09	78	0019D	ASHL	#9, -4(R3), -(SP)		
00000000G	00		01	FB	001A2	CALLS	#1, GET_VM		
	FC	A4	50	D0	001A9	MOVL	R0, -4(R4)		
			7E	7C	001AD	CLRQ	-(SP)	1484	
			7E	D4	001AF	CLRL	-(SP)		
			52	DD	001B1	PUSHL	BITMAP_OFFSET		
		00000000'FF	48	DE	001B3	MOVAL	@FAST_IMAP_SIZE[RVN], R0		
7E	FC	A0	09	78	001BB	ASHL	#9, -4(R0), -(SP)		
		00000000'FF	48	DE	001C0	MOVAL	@FAST_IMAP[RVN], R0		
		FC	A0	DD	001C8	PUSHL	-4(R0)		
			7E	7C	001CB	CLRQ	-(SP)		
		40	AE	9F	001CD	PUSHAB	IOSB		
			31	DD	001D0	PUSHL	#49		
		00000000'	EF	DD	001D2	PUSHL	INPUT_CHAN		
			7E	D4	001D8	CLRL	-(SP)		
00000000G	00		0C	FB	001DA	CALLS	#12, SYSSQIOW		
	5A		50	D0	001E1	MOVL	R0, STATUS		
	07		5A	E9	001E4	BLBC	STATUS, 11\$	1485	
	5A	20	AE	3C	001E7	MOVZWL	IOSB, STATUS		
	1B		5A	E8	001EB	BLBS	STATUS, 12\$	1486	
		0500	8F	BB	001EE	11\$:	PUSHR	#*M<R8,R10>	1488
7E	00000000'	EF	10	C1	001F2	ADDL3	#16, INPUT_QUAL, -(SP)		
			02	DD	001FA	PUSHL	#2		



00000000G	00	00000000G	8F	DD	001FC	PUSHL	#BACKUP\$ PROCINDEX		
			05	FB	00202	CALLS	#5, LIB\$SIGNAL		
			7E	7C	00209	12\$: CLRQ	-(SP)	1499	
			7E	D4	0020B	CLRL	-(SP)		
7E	FC	50	00000000'	FF48	DE	0020D	MOVAL	@FAST_HDR_OFFSET[RVN], R0	
		A0		01	C1	00215	ADDL3	#1, -4(R0), -(SP)	
		7E	0200	8F	3C	0021A	MOVZWL	#512, -(SP)	
7E	00000000'	EF	00000200	8F	C1	0021F	ADDL3	#512, FAST_BUFFER, -(SP)	
			40	7E	7C	0022B	CLRQ	-(SP)	
				AE	9F	0022D	PUSHAB	IOSB	
				31	DD	00230	PUSHL	#49	
			00000000'	EF	DD	00232	PUSHL	INPUT_CHAN	
				7E	D4	00238	CLRL	-(SP)	
00000000G	00			0C	FB	0023A	CALLS	#12, SYSSQIOW	
	5A			50	D0	00241	MOVL	R0, STATUS	
	07			5A	E9	00244	BLBC	STATUS, 13\$	
	5A	20		AE	3C	00247	MOVZWL	IOSB, STATUS	
	1B			5A	E8	0024B	BLBS	STATUS, 14\$	
		0500		8F	BB	0024E	13\$: PUSHR	#^M<R8,R10>	
7E	00000000'	EF		10	C1	00252	ADDL3	#16, INPUT_QUAL, -(SP)	
				02	DD	0025A	PUSHL	#2	
			00000000G	8F	DD	0025C	PUSHL	#BACKUP\$ PROCINDEX	
				05	FB	00262	CALLS	#5, LIB\$SIGNAL	
				57	D4	00269	14\$: CLRL	EOF	
			50	00000000'	EF	D0	0026B	MOVL	FAST_BUFFER, R0
			02	00000000'	EF	91	00272	CMPB	FAST_STRUCLÉV, #2
				0C	12	00279	BNEQ	15\$	
50	021C	C0		10	9C	0027B	ROTL	#16, 540(R0), R0	
		57	FF	A0	9E	00281	MOVAB	-1(R0), EOF	
				0B	11	00285	BRB	16\$	
		02	0C	A0	91	00287	15\$: CMPB	12(R0), #2	
				05	12	0028B	BNEQ	16\$	
57	10	AE		10	9C	0028D	ROTL	#16, STATBLK+4, EOF	
		53	00000000'	FF48	DE	00292	16\$: MOVAL	@FAST_IMAP_SIZE[RVN], R3	
53	FC	A3		07	78	0029A	ASHL	#7, -4(R3), R3	
		52		53	D0	0029F	MOVL	R3, J	
				39	11	002A2	BRB	19\$	
		50	00000000'	FF48	DE	002A4	17\$: MOVAL	@FAST_IMAP[RVN], R0	
		50	FC B042	D0	002AC		MOVL	@-4(R0)[J], R0	
				2A	13	002B1	BEQL	19\$	
53		52		05	78	002B3	ASHL	#5, J, R3	
				50	DD	002B7	PUSHL	R0	
		00		01	FB	002B9	CALLS	#1, LEFT ONE	
51	00000000G	53		50	C1	002C0	ADDL3	R0, R3, R1	
		50	00000000'	FF48	DE	002C4	MOVAL	@FAST_HDR_OFFSET[RVN], R0	
		51	FC	A0	C0	002CC	ADDL2	-4(R0), RT	
		57		51	D1	002D0	CMP	R1, EOF	
				03	1E	002D3	BGEQU	18\$	
		51		57	D0	002D5	MOVL	EOF, R1	
		57		51	D0	002D8	18\$: MOVL	R1, EOF	
				03	11	002DB	BRB	20\$	
		C4		52	F4	002DD	19\$: SOBGEQ	J, 17\$	
74	00000000'	EF		03	E1	002E0	20\$: BBC	#3, QUAL+10, 23\$	
				7E	7C	002E8	CLRQ	-(SP)	
		7E		01	7D	002EA	MOVQ	#1, -(SP)	
		7E	0200	8F	3C	002ED	MOVZWL	#512, -(SP)	
7E	00000000'	EF	00000200	8F	C1	002F2	ADDL3	#512, FAST_BUFFER, -(SP)	

			40	7E	7C	002FE	CLRQ	-(SP)	
				AE	9F	00300	PUSHAB	IOSB	
				31	DD	00303	PUSHL	#49	
			00000000'	EF	DD	00305	PUSHL	INPUT_CHAN	
				7E	D4	0030B	CLRL	-(SP)	
				0C	FB	0030D	CALLS	#12, SYSSQIOW	
				50	D0	00314	MOVL	R0, STATUS	
				5A	E9	00317	BLBC	STATUS, 21\$	1545
			20	AE	3C	0031A	MOVZWL	IOSB, STATUS	
				5A	E8	0031E	BLBS	STATUS, 22\$	1546
			0500	8F	BB	00321	PUSHR	#^M<R8,R10>	1548
				10	C1	00325	ADDL3	#16, INPUT_QUAL, -(SP)	
				02	DD	0032D	PUSHL	#2	
			00000000G	8F	DD	0032F	PUSHL	#BACKUP\$ PROCINDEX	
				05	FB	00335	CALLS	#5, LIB\$SIGNAL	
			00000000'FF	48	DE	0033C	MOVAL	@FAST_BOOT_LBN[RVN], R0	1553
			FC	A0	01	CE	MNEGL	#1, -4(R0)	
				51	DO	00348	MOVL	FAST_BUFFER, R1	1554
			0205	C1	95	0034F	TSTB	517(R1)	
				07	12	00353	BNEQ	23\$	
				10	9C	00355	ROTL	#16, 516(R1), -4(R0)	1556
				7E	7C	0035C	CLRQ	-(SP)	1564
				7E	7C	0035E	CLRQ	-(SP)	
				7E	7C	00360	CLRQ	-(SP)	
				7E	7C	00362	CLRQ	-(SP)	
			7E	34	7D	00364	MOVQ	#52, -(SP)	
			00000000'	EF	DD	00367	PUSHL	INPUT_CHAN	
				7E	D4	0036D	CLRL	-(SP)	
			00000000G	00	0C	FB	CALLS	#12, SYSSQIOW	
			03 00000000'	EF	03	E0	BBS	#3, QUAL+10, 24\$	1567
				0107	31	0037E	BRW	30\$	
				00	2C	00381	MOVCS	#0, (SP), #0, #64, FIB	1573
				AE		00388			
			30	56	D0	0038A	MOVL	ACCTL, FIB	1574
			34	8F	D0	0038E	MOVL	#131074, FIB+4	1575
			38	58	B0	00396	MOVW	RVN, FIB+8	1577
				7E	7C	0039A	CLRQ	-(SP)	1582
				7E	7C	0039C	CLRQ	-(SP)	
				7E	D4	0039E	CLRL	-(SP)	
			3C	AE	9F	003A0	PUSHAB	FIB_DESC	
				7E	7C	003A3	CLRQ	-(SP)	
			40	AE	9F	003A5	PUSHAB	IOSB	
			7E	8F	9A	003A8	MOVZBL	#114, -(SP)	
			00000000'	EF	DD	003AC	PUSHL	INPUT_CHAN	
				7E	D4	003B2	CLRL	-(SP)	
			00000000G	00	0C	FB	CALLS	#12, SYSSQIOW	
				5A	D0	003BB	MOVL	R0, STATUS	
				07	5A	E9	BLBC	STATUS, 25\$	1583
				5A	3C	003C1	MOVZWL	IOSB, STATUS	
			20	5A	E8	003C5	BLBS	STATUS, 26\$	1584
			0500	8F	BB	003C8	PUSHR	#^M<R8,R10>	1586
				10	C1	003CC	ADDL3	#16, INPUT_QUAL, -(SP)	
			7E 00000000'	EF	02	DD	PUSHL	#2	
					8F	DD	PUSHL	#BACKUP\$ PROCINDEX	
			00000000G	00	05	FB	CALLS	#5, LIB\$SIGNAL	
				7E	7C	003E3	CLRQ	-(SP)	1597
				01	7D	003E5	MOVQ	#1, -(SP)	



7E 00000000'	7E 0200	8F 3C 003E8	MOVZWL	#512, -(SP)	
	EF 00000400	8F C1 003ED	ADDL3	#1024, FAST_BUFFER, -(SP)	
		7E 7C 003F9	CLRQ	-(SP)	
	40	AE 9F 003FB	PUSHAB	IOSB	
		31 DD 003FE	PUSHL	#49	
	00000000'	EF DD 00400	PUSHL	INPUT_CHAN	
		7E D4 00406	CLRL	-(SP)	
00000000G	00	OC FB 00408	CALLS	#12, SYSSQIOW	
	5A	50 D0 0040F	MOVL	R0, STATUS	
	07	5A E9 00412	BLBC	STATUS, 27\$	1598
	5A 20	AE 3C 00415	MOVZWL	IOSB, STATUS	
	1B	5A E8 00419	BLBS	STATUS, 28\$	1599
7E 00000000'	EF 0500	8F BB 0041C	PUSHR	#^M<R8,R10>	1601
		10 C1 00420	ADDL3	#16, INPUT_QUAL, -(SP)	
		02 DD 00428	PUSHL	#2	
00000000G	00 00000000G	8F DD 0042A	PUSHL	#BACKUP\$ PROCINDEX	
		05 FB 00430	CALLS	#5, LIB\$SIGNAL	
		7E 7C 00437	CLRQ	-(SP)	1608
		7E 7C 00439	CLRQ	-(SP)	
		7E 7C 0043B	CLRQ	-(SP)	
		7E 7C 0043D	CLRQ	-(SP)	
	7E	34 7D 0043F	MOVQ	#52, -(SP)	
	00000000'	EF DD 00442	PUSHL	INPUT_CHAN	
		7E D4 00448	CLRL	-(SP)	
00000000G	00	OC FB 0044A	CALLS	#12, SYSSQIOW	
	0B 00000000'	EF E9 00451	BLBC	QUAL+14, 29\$	1613
58 00000000'	EF 08	00 ED 00458	CMPZV	#0, #8, QUAL+79, RVN	
		25 12 00461	BNEQ	30\$	
	50 00000000'	FF 48 DE 00463	MOVAL	@FAST_HDR_OFFSET[RVN], R0	1619
	7E 57 FC	A0 C3 0046B	SUBL3	-4(R0), EOF, -(SP)	
	50 00000000'	EF D0 00470	MOVL	FAST_BUFFER, R0	1618
		0400	PUSHAB	1024(R0)	
		0200	PUSHAB	512(R0)	1617
		50 DD 0047F	PUSHL	R0	1616
0040 8F	00 00000000G	04 FB 00481	CALLS	#4, VOLUME_ATTRIBUTES	
	6E	00 2C 00488	MOVCS	#0, (SP), #0, #64, FIB	1625
	30	AE 0048F			
	34 AE 00010001	56 D0 00491	MOVL	ACCTL, FIB	1626
	38 AE	8F D0 00495	MOVL	#65537, FIB+4	1627
		58 B0 0049D	MOVW	RVN, FIB+8	1629
		7E 7C 004A1	CLRQ	-(SP)	1634
		7E 7C 004A3	CLRQ	-(SP)	
		7E D4 004A5	CLRL	-(SP)	
	3C	AE 9F 004A7	PUSHAB	FIB_DESC	
		7E 7C 004AA	CLRQ	-(SP)	
	40	AE 9F 004AC	PUSHAB	IOSB	
	7E 72	8F 9A 004AF	MOVZBL	#114, -(SP)	
	00000000'	EF DD 004B3	PUSHL	INPUT_CHAN	
		7E D4 004B9	CLRL	-(SP)	
00000000G	00	OC FB 004BB	CALLS	#12, SYSSQIOW	
	5A	50 D0 004C2	MOVL	R0, STATUS	
	07	5A E9 004C5	BLBC	STATUS, 31\$	1635
	5A 20	AE 3C 004C8	MOVZWL	IOSB, STATUS	
	1B	5A E8 004CC	BLBS	STATUS, 32\$	1636
	0500	8F BB 004CF	PUSHR	#^M<R8,R10>	1638
7E 00000000'	EF	10 C1 004D3	ADDL3	#16, INPUT_QUAL, -(SP)	
		02 DD 004DB	PUSHL	#2	



	00000000G	00	00000000G	8F	DD	004DD	PUSHL	#BACKUP\$ PROCINDEX		
				05	FB	004E3	CALLS	#5, LIB\$SIGNAL		
				EF	94	004EA	CLRB	DIR STATUS		1644
				FF	DE	004F0	MOVAL	@FAST HDR OFFSET[RVN], R0		1645
59	FC	50	00000000'	01	C1	004F8	ADDL3	#1, -4(R0), VBN		
		6E		A7	9E	004FD	MOVAB	1(R7), (SP)		1655
		57		59	D1	00501	CMPL	VBN, EOF		1646
				03	1B	00504	BLEQU	34\$		
				01B4	31	00506	BRW	58\$		
50		6E		59	C3	00509	SUBL3	VBN, (SP), R0		1655
	00000040	8F		50	D1	0050D	CMPL	R0, #64		
				04	1B	00514	BLEQU	35\$		
		50		8F	9A	00516	MOVZBL	#64, R0		
		5B		50	D0	0051A	MOVL	R0, READ_COUNT		
				7E	7C	0051D	CLRQ	-(SP)		1662
				7E	D4	0051F	CLRL	-(SP)		
7E		5B		59	DD	00521	PUSHL	VBN		
				09	78	00523	ASHL	#9, READ COUNT, -(SP)		
			00000000'	EF	DD	00527	PUSHL	FAST_BUFFER		
				7E	7C	0052D	CLRQ	-(SP)		
				40	AE	9F	0052F	PUSHAB	IOSB	
				31	DD	00532	PUSHL	#49		
			00000000'	EF	DD	00534	PUSHL	INPUT_CHAN		
				7E	D4	0053A	CLRL	-(SP)		
	00000000G	00		0C	FB	0053C	CALLS	#12, SYSSQIOW		
		5A		50	D0	00543	MOVL	R0, STATUS		
		07		5A	E9	00546	BLBC	STATUS, 36\$		1663
		5A		AE	3C	00549	MOVZWL	IOSB, STATUS		
		4E		5A	E8	0054D	BLBS	STATUS, 40\$		1669
		56		01	CE	00550	MNEGL	#1, XVBN		1672
				45	11	00553	BRB	39\$		
50		56		09	78	00555	ASHL	#9, XVBN, R0		1678
52		50	00000000'	EF	C1	00559	ADDL3	FAST_BUFFER, R0, HEADER		
				7E	7C	00561	CLRQ	-(SP)		1685
				7E	D4	00563	CLRL	-(SP)		
				6649	9F	00565	PUSHAB	(XVBN)[VBN]		
		7E	0200	8F	3C	00568	MOVZWL	#512, -(SP)		
				52	DD	0056D	PUSHL	HEADER		
				7E	7C	0056F	CLRQ	-(SP)		
				40	AE	9F	00571	PUSHAB	IOSB	
				31	DD	00574	PUSHL	#49		
			00000000'	EF	DD	00576	PUSHL	INPUT_CHAN		
				7E	D4	0057C	CLRL	-(SP)		
	00000000G	00		0C	FB	0057E	CALLS	#12, SYSSQIOW		
		5A		50	D0	00585	MOVL	R0, STATUS		
		07		5A	E9	00588	BLBC	STATUS, 38\$		1686
		5A		AE	3C	0058B	MOVZWL	IOSB, STATUS		
		08		5A	E8	0058F	BLBS	STATUS, 39\$		1687
0200	8F	00		00	2C	00592	MOVCS	#0, (SP), #0, #512, (HEADER)		1689
				62		00599				
		56		5B	F2	0059A	AOBLSS	READ_COUNT, XVBN, 37\$		1672
B7		52	00000000'	EF	D0	0059E	MOVL	FAST_BUFFER, HEADER		1697
		54		01	CE	005A5	MNEGL	#1, XVBN		1698
				0102	31	005A8	BRW	55\$		
51		59		54	C1	005AB	ADDL3	XVBN, VBN, R1		1708
		50	00000000'	FF	DE	005AF	MOVAL	@FAST HDR OFFSET[RVN], R0		
53		51	FC	A0	C3	005B7	SUBL3	-4(R0), RT, FILE_NUMBER		



50	53	04	AE	53	B0	005BC	MOVW	FILE_NUMBER, FILE_ID	1709
		08		10	EF	005C0	EXTZV	#16, #8, FILE_NUMBER, R0	1710
		09	AE	50	90	005C5	MOVB	R0, FILE_ID+5	
		02	00000000'	EF	91	005C9	CMPB	FAST_STRUCLEV, #2	1711
				07	12	005D0	BNEQ	42\$	
		06	AE	A2	B0	005D2	MOVW	10(HEADER), FILE_ID+2	1712
				05	11	005D7	BRB	43\$	
		06	AE	A2	B0	005D9	MOVW	4(HEADER), FILE_ID+2	1713
		08	AE	58	90	005DE	MOVB	RVN, FILE_ID+4	1714
	25	00000000'	EF	03	E1	005E2	BBC	#3, QUAL+T0, 45\$	1720
				54	D5	005EA	TSTL	XVBN	1721
				21	12	005EC	BNEQ	45\$	
		0B	00000000'	EF	E9	005EE	BLBC	QUAL+14, 44\$	1722
		08		00	ED	005F5	CMPZV	#0, #8, QUAL+79, RVN	
				0F	12	005FE	BNEQ	45\$	
			0108	8F	BB	00600	PUSHR	#*M<R3,R8>	1724
			0804	8F	BB	00604	PUSHR	#*M<R2,R11>	
		00000000G	00	04	FB	00608	CALLS	#4, GEN_FID_RECORD	
				AE	9F	0060F	PUSHAB	FILE_ID	1732
				52	DD	00612	PUSHL	HEADER	
		0000V	CF	02	FB	00614	CALLS	#2, VERIFY_HEADER	
			5A	50	D0	00619	MOVL	R0, STATUS	
			01	00000000'	EF	91	CMPB	FAST_STRUCLEV, #1	1734
				13	12	00623	BNEQ	46\$	
			50	00000000'FF	48	DE	MOVAL	@FAST_IMAP[RVN], R0	1735
			51	FF	A3	9E	MOVAB	-1(R3), R1	
	02	FC	B0	51	E0	00631	BBS	R1, @-4(R0), 46\$	
				5A	D4	00636	CLRL	STATUS	1737
			50	00000000'FF	48	DE	MOVAL	@FAST_IMAP[RVN], R0	1743
				53	D7	00640	DECL	R3	
	00	FC	B0	53	E5	00642	BBCC	R3, @-4(R0), 47\$	
			5E	5A	E9	00647	BLBC	STATUS, 54\$	1746
			51	01	A2	9A	MOVZBL	1(HEADER), R1	1753
			51	6241	3E	0064E	MOVAB	(HEADER)[R1], MAP_AREA	
				55	D4	00652	CLRL	R5	1756
			02	00000000'	EF	91	CMPB	FAST_STRUCLEV, #2	
				07	12	0065B	BNEQ	48\$	
				55	D6	0065D	INCL	R5	
			04	A2	B5	0065F	TSTW	4(HEADER)	1757
				02	11	00662	BRB	49\$	
				61	95	00664	TSTB	(MAP_AREA)	1758
				40	12	00666	BNEQ	54\$	
	00	FC	B0	53	E2	00668	BBSS	R3, @-4(R0), 50\$	1769
			06	55	E9	0066D	BLBC	R5, 51\$	1774
			50	A2	9E	00670	MOVAB	52(R2), FCH	1775
				04	11	00674	BRB	52\$	
			50	A2	9E	00676	MOVAB	12(R2), FCH	1776
				60	B5	0067A	TSTW	(FCH)	1785
				1D	19	0067C	BLSS	53\$	
				52	DD	0067E	PUSHL	HEADER	1790
		00000000G	00	01	FB	00680	CALLS	#1, INIT_ATTR	
	19	00000000'	EF	03	E0	00687	BBS	#3, QUAL+T0, 54\$	1791
				01	DD	0068F	PUSHL	#1	1793
		00000000G	00	01	FB	00691	CALLS	#1, SELECT_INPUT_FILE	
			0D	50	E8	00698	BLBS	R0, 54\$	
			50	00000000'FF	48	DE	MOVAL	@FAST_IMAP[RVN], R0	1797
	00	FC	B0	53	E4	006A3	BBSC	R3, @-4(R0), 54\$	



02	52	0200	C2	9E	006A8	54\$:	MOVAB	512(R2), HEADER	1802	
	54		5B	F2	006AD	55\$:	AOBLSS	READ_COUNT, XVBN, 56\$	1698	
			03	11	006B1		BRB	57\$		
			FEF5	31	006B3	56\$:	BRW	41\$		
	59	40	A9	9E	006B6	57\$:	MOVAB	64(R9), VBN	1806	
			FE44	31	006BA		BRW	33\$	1646	
			7E	7C	006BD	58\$:	CLRQ	-(SP)	1814	
			7E	7C	006BF		CLRQ	-(SP)		
			7E	7C	006C1		CLRQ	-(SP)		
			7E	7C	006C3		CLRQ	-(SP)		
	7E		34	7D	006C5		MOVQ	#52, -(SP)		
		00000000'	EF	DD	006C8		PUSHL	INPUT_CHAN		
			7E	D4	006CE		CLRL	-(SP)		
	00000000G	00	0C	FB	006D0		CALLS	#12, SYSSQIOW		
			58	D6	006D7		INCL	RVN	1817	
58 00000000' EF	08		00	ED	006D9		CMPZV	#0, #8, COM_I_SETCOUNT, RVN	1819	
			03	1F	006E2		BLSSU	59\$		
			F950	31	006E4		BRW	1\$		
		00000000'	EF	DD	006E7	59\$:	PUSHL	FAST_BUFFER	1824	
		00000000'	EF	DD	006ED		PUSHL	FAST_BUFFER_SIZE		
	00000000G	00	02	FB	006F3		CALLS	#2, FREE_VM		
		00000000'	EF	7C	006FA		CLRQ	FAST_BUFFER	1825	
	00000000'	EF	40	8F	8A	00700	BICB2	#64, COM_FLAGS	1826	
		53 00000000'	EF	9A	00708		MOVZBL	COM_I_SETCOUNT, R3	1831	
			52	D4	0070F		CLRL	RVN		
			07	11	00711		BRB	61\$		
			52	DD	00713	60\$:	PUSHL	RVN		
	0000V	CF	01	FB	00715		CALLS	#1, DIR_SCAN		
F5		52	53	F3	0071A	61\$:	AOBLEQ	R3, RVN, 60\$		
	00000000'	EF	0200	8F	3C	0071E	MOVZWL	#512, FAST_BUFFER_SIZE	1836	
		7E	0200	8F	3C	00727	MOVZWL	#512, -(SP)	1837	
		00000000G	00	01	FB	0072C	CALLS	#1, GET_VM		
		00000000'	EF	50	D0	00733	MOVL	R0, FAST_BUFFER		
03 00000000'		EF	03	E0	0073A		BBS	#3, QUAL+10, 62\$	1842	
			01CA	31	00742		BRW	75\$		
		59 00000000'	EF	9A	00745	62\$:	MOVZBL	COM_I_SETCOUNT, R9	1845	
			56	D4	0074C		CLRL	RVN		
			01B8	31	0074E		BRW	74\$		
		50 00000000'	FF46	DE	00751	63\$:	MOVAL	@FAST_IMAP_SIZE[RVN], R0	1850	
57	FC	A0	0C	78	00759		ASHL	#12, =4(R0), R7		
			58	D4	0075E		CLRL	FILE_NUMBER		
			01A0	31	00760	64\$:	BRW	73\$		
		50 00000000'	FF46	DE	00763	65\$:	MOVAL	@FAST_IMAP[RVN], R0	1852	
		51	FF	A8	9E	0076B	MOVAB	-1(R8), R1		
		EC	B0	51	E1	0076F	BBC	R1, @-4(R0), 64\$		
0040 8F		00	6E	00	2C	00774	MOVCS	#0, (SP), #0, #64, FIB	1864	
			30	AE	0077B					
		30	AE	00200000	8F	D0	0077D	MOVL	#2097152, FIB	1865
		34	AE	00010001	8F	D0	00785	MOVL	#65537, FIB+4	1866
		38	AE		56	B0	0078D	MOVW	RVN, FIB+8	1868
			7E	7C	00791		CLRQ	-(SP)	1873	
			7E	7C	00793		CLRQ	-(SP)		
			7E	D4	00795		CLRL	-(SP)		
		3C	AE	9F	00797		PUSHAB	FIB_DESC		
			7E	7C	0079A		CLRQ	-(SP)		
		40	AE	9F	0079C		PUSHAB	IOSB		
		7E	72	8F	9A	0079F	MOVZBL	#114, -(SP)		



		00000000'	EF DD 007A3	PUSHL INPUT_CHAN	
			7E D4 007A9	CLRL -(SP)	
00000000G	00		OC FB 007AB	CALLS #12, SYSSQIOW	
	5A		50 D0 007B2	MOVL R0, STATUS	
	07		5A E9 007B5	BLBC STATUS, 66\$	1874
	5A	20	AE 3C 007B8	MOVZWL IOSB, STATUS	
	1B		5A E8 007BC	BLBS STATUS, 67\$	1875
		0440	8F BB 007BF	PUSHR #*M<R6,R10>	1877
7E 00000000'	EF		10 C1 007C3	ADDL3 #16, INPUT_QUAL, -(SP)	
			02 DD 007CB	PUSHL #2	
		00000000G	8F DD 007CD	PUSHL #BACKUP\$ PROCINDEX	
00000000G	00		05 FB 007D3	CALLS #5, LIB\$SIGNAL	
			7E 7C 007DA	CLRL -(SP)	1888
			7E D4 007DC	CLRL -(SP)	
	50	00000000'	FF46 DE 007DE	MOVAL @FAST_HDR_OFFSET[RVN], R0	
		FC B048	9F 007E6	PUSHAB @-4(R0)[FILE_NUMBER]	
	7E	0200	8F 3C 007EA	MOVZWL #512, -(SP)	
		00000000'	EF DD 007EF	PUSHL FAST_BUFFER	
			7E 7C 007F5	CLRL -(SP)	
		40	AE 9F 007F7	PUSHAB IOSB	
			31 DD 007FA	PUSHL #49	
		00000000'	EF DD 007FC	PUSHL INPUT_CHAN	
			7E D4 00802	CLRL -(SP)	
00000000G	00		OC FB 00804	CALLS #12, SYSSQIOW	
	5A		50 D0 0080B	MOVL R0, STATUS	
	07		5A E9 0080E	BLBC STATUS, 68\$	1889
	5A	20	AE 3C 00811	MOVZWL IOSB, STATUS	
	1B		5A E8 00815	BLBS STATUS, 69\$	1890
		0440	8F BB 00818	PUSHR #*M<R6,R10>	1892
7E 00000000'	EF		10 C1 0081C	ADDL3 #16, INPUT_QUAL, -(SP)	
			02 DD 00824	PUSHL #2	
		00000000G	8F DD 00826	PUSHL #BACKUP\$ PROCINDEX	
00000000G	00		05 FB 0082C	CALLS #5, LIB\$SIGNAL	
			7E 7C 00833	CLRL -(SP)	1899
			7E 7C 00835	CLRL -(SP)	
			7E 7C 00837	CLRL -(SP)	
			7E 7C 00839	CLRL -(SP)	
	7E		34 7D 0083B	MOVQ #52, -(SP)	
		00000000'	EF DD 0083E	PUSHL INPUT_CHAN	
			7E D4 00844	CLRL -(SP)	
00000000G	00		OC FB 00846	CALLS #12, SYSSQIOW	
	50	00000000'	EF D0 0084D	MOVL INPUT_NAM, R0	1904
	24	A0	58 B0 00854	MOVW FILE_NUMBER, 36(R0)	
			10 EF 00858	EXTZV #16, #8, FILE_NUMBER, R1	1905
	29	A0	51 90 0085D	MOVB R1, 41(R0)	
	28	A0	56 90 00861	MOVB RVN, 40(R0)	1906
		2A	A0 D4 00865	CLRL 42(R0)	1907
		2E	A0 B4 00868	CLRW 46(R0)	1909
	51	00000000'	EF D0 0086B	MOVL FAST_BUFFER, R1	1921
	02	00000000'	EF 91 00872	CMPB FAST_STRUCLÉV, #2	1915
			21 12 00879	BNEQ 71\$	
	26	A0	0A A1 B0 0087B	MOVW 10(R1), 38(R0)	1921
			61 9A 00880	MOVZBL (R1), R0	1923
		6140	3E 00883	MOVAW (R1)[R0], NAME_ADDRESS	1924
			14 D0 00887	MOVL #20, NAME_LENGTH	1925
			20 3A 0088A	LOCC #32, #20, -(NAME_ADDRESS)	1926
62		14	02 12 0088E	BNEQ 70\$	



				51	D4	00890	CLRL	R1		
				51	D5	00892	TSTL	P		1927
				23	13	00894	BEQL	72\$		
53		51		52	C3	00896	SUBL3	NAME_ADDRESS, P, NAME_LENGTH		
				1D	11	0089A	BRB	72\$		1915
	26	A0	04	A1	B0	0089C	MOVW	4(R1), 38(R0)		1931
		52	04	AE	9E	008A1	MOVAB	FILENAME, NAME_ADDRESS		1932
			04	AE	9F	008A5	PUSHAB	FILENAME		1933
		50		61	9A	008A8	MOVZBL	(R1), R0		1934
			FA	A140	3F	008AB	PUSHAW	-6(R1)[R0]		1935
	00000000G	00		02	FB	008AF	CALLS	#2, MAKE_STRING		
		53		50	3C	008B6	MOVZWL	R0, NAME_LENGTH		1936
	18	AE	FF	8F	9A	008B9	MOVZBL	#255, RSA_DESC		1938
		50	00000000'	EF	D0	008BE	MOVL	INPUT_NAM, R0		1939
	1C	AE	04	A0	D0	008C5	MOVL	4(R0), RSA_DESC+4		
				52	DD	008CA	PUSHL	NAME_ADDRESS		1945
				53	DD	008CC	PUSHL	NAME_LENGTH		
7E	00000000'	EF		10	C1	008CE	ADDL3	#16, INPUT_QUAL, -(SP)		
			24	AE	9F	008D6	PUSHAB	RSA_DESC		
			28	AE	9F	008D9	PUSHAB	RSA_DESC		
			F718	CF	9F	008DC	PUSHAB	P.AAB		
	00000000G	00		06	FB	008E0	CALLS	#6, SYSSFAO		
		50	00000000'	EF	D0	008E7	MOVL	INPUT_NAM, R0		1946
	03	A0	18	AE	90	008EE	MOVB	RSA_DESC, 3(R0)		
				50	DD	008F3	PUSHL	R0		1947
	00000000G	00		01	FB	008F5	CALLS	#1, INIT_NAMEBLOCK		
	00000000G	00		00	FB	008FC	CALLS	#0, SAVE_ONE_FILE		1952
				57	F1	00903	ACBL	R7, #1, FILE_NUMBER, 65\$		1850
FE5A				59	F1	00909	ACBL	R9, #1, RVN, 63\$		1845
FE42					EF	DD	0090F	75\$:		1961
			00000000'	EF	DD	00915	PUSHL	FAST_BUFFER		
			00000000'	EF	DD	00915	PUSHL	FAST_BUFFER_SIZE		
	00000000G	00		02	FB	0091B	CALLS	#2, FREE_VM		
			00000000'	EF	7C	00922	CLRQ	FAST_BUFFER		1962
		54	00000000'	EF	9A	00928	MOVZBL	COM_I_SETCOUNT, R4		1967
		53	00000000'	EF	D0	0092F	MOVL	FAST_IMAP, R3		1969
				52	D4	00936	CLRL	RVN		
				23	11	00938	BRB	77\$		
			FC	A342	DD	0093A	76\$:			
		50	00000000'	FF42	DE	0093E	PUSHL	-4(R3)[RVN]		
7E		A0		09	78	00946	MOVAL	@FAST_IMAP_SIZE[RVN], R0		
	FC	00		02	FB	0094B	ASHL	#9, -4(R0), -(SP)		
	00000000G			53	00000000'	EF	D0	00952		
			FC	A342	D4	00959	77\$:			
		52		54	F3	0095D	AOBLEQ	R4, RVN, 76\$		1967
D9		52	00000000'	EF	9E	00961	MOVAB	FAST_VOL_BEG, R2		1976
		53	00000000'	EF	9E	00968	MOVAB	FAST_VOL_END-4, R3		
				16	11	0096F	BRB	79\$		
				62	DD	00971	78\$:			1978
		50	00000000'	EF	9A	00973	PUSHL	(A)		
7E		50		02	78	0097A	MOVZBL	COM_I_SETCOUNT, R0		
	00000000G	00		02	FB	0097E	ASHL	#2, -R0, -(SP)		
				82	D4	00985	CALLS	#2, FREE_VM		1979
				52	D1	00987	CLRL	(A)+		1976
		53		E5	1B	0098A	CMPL	A, R3		
					04	0098C	BLEQU	78\$		
							RET			1981

; Routine Size: 2445 bytes; Routine Base: CODE + 0018



FASTSCAN  
V04-000

Fast file scan  
FAST\_FILE\_SCAN - fast file scan main routine

E 5  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 34  
(3)

```
: 880 1982 1 %SBTTL 'SLOW FILE SCAN - slow file scan main routine'
: 881 1983 1 GLOBAL ROUTINE SLOW_FILE_SCAN: NOVALUE=
: 882 1984 1
: 883 1985 1 ++
: 884 1986 1
: 885 1987 1 FUNCTIONAL DESCRIPTION:
: 886 1988 1 This routine is the driver for the slow file scan.
: 887 1989 1
: 888 1990 1 INPUT PARAMETERS:
: 889 1991 1 NONE
: 890 1992 1
: 891 1993 1 IMPLICIT INPUTS:
: 892 1994 1 NONE
: 893 1995 1
: 894 1996 1 OUTPUT PARAMETERS:
: 895 1997 1 NONE
: 896 1998 1
: 897 1999 1 IMPLICIT OUTPUTS:
: 898 2000 1 NONE
: 899 2001 1
: 900 2002 1 ROUTINE VALUE:
: 901 2003 1 NONE
: 902 2004 1
: 903 2005 1 SIDE EFFECTS:
: 904 2006 1 NONE
: 905 2007 1
: 906 2008 1 --
: 907 2009 1
: 908 2010 2 BEGIN
: 909 2011 2 DIR_SCAN(1);
: 910 2012 1 END;
```

```
0000 00000
01 DD 00002
01 FB 00004
04 00009
```

```
.ENTRY SLOW_FILE_SCAN, Save nothing
PUSHL #1
CALLS #1, DIR_SCAN
RET
```

```
: 1983
: 2011
: 2012
```

; Routine Size: 10 bytes, Routine Base: CODE + 09A5



```
912 2013 1 %SBTTL 'READ_HOMEBLOCK - read home block from index file'
913 2014 1 ROUTINE READ_HOMEBLOCK(RVN,BUFFER): NOVALUE=
914 2015 1
915 2016 1 !++
916 2017 1
917 2018 1 FUNCTIONAL DESCRIPTION:
918 2019 1 This routine reads the first good home block of the currently open
919 2020 1 index file into the buffer supplied.
920 2021 1
921 2022 1 INPUT PARAMETERS:
922 2023 1 RVN - Relative volume number.
923 2024 1 BUFFER - Pointer to buffer.
924 2025 1
925 2026 1 IMPLICIT INPUTS:
926 2027 1 NONE
927 2028 1
928 2029 1 OUTPUT PARAMETERS:
929 2030 1 NONE
930 2031 1
931 2032 1 IMPLICIT OUTPUTS:
932 2033 1 BUFFER - Contains a valid home block.
933 2034 1 COM_1_SETCOUNT - Count of volumes in volume set.
934 2035 1 FAST_STRUCLEV - Structure level (1 or 2) of the volume set.
935 2036 1
936 2037 1 ROUTINE VALUE:
937 2038 1 NONE
938 2039 1
939 2040 1 SIDE EFFECTS:
940 2041 1 NONE
941 2042 1
942 2043 1 !--
943 2044 1
944 2045 2 BEGIN
945 2046 2 MAP
946 2047 2 BUFFER: REF BBLOCK; ! Pointer to buffer
947 2048 2
948 2049 2 LOCAL STATUS, ! General status value
949 2050 2 IOSB: VECTOR[4,WORD], ! I/O status block
950 2051 2 OLD_STATUS; ! Save status for error message
951 2052 2
952 2053 2
953 2054 2 ! We keep reading until we get a block that reads without errors and looks
954 2055 2 ! like a home block. Track any error status for the eventual error message.
955 2056 2
956 2057 2 OLD STATUS = $$$ ABORT;
957 2058 2 INCR VBN FROM 2 TO 100 DO
958 2059 2 BEGIN
959 2060 2 STATUS = $QIOW(
960 2061 2 FUNC=IOS READVBLK,
961 2062 2 CHAN=.INPUT_CHAN,
962 2063 2 IOSB=IOSB,
963 2064 2 P1=.BUFFER,
964 2065 2 P2=512,
965 2066 2 P3=.VBN);
966 2067 2 IF .STATUS THEN STATUS = .IOSB[0];
967 2068 2
968 2069 2 IF NOT .STATUS
```

```

: 969 2070 3 THEN
: 970 2071 3 OLD_STATUS = .STATUS
: 971 2072 3 ELSE
: 972 2073 3 IF
: 973 2074 3 .BUFFER[HM2$B_STRUCLEV] EQL 2 AND
: 974 2075 3 .BUFFER[HM2$W_HOMEBVN] EQL .VBN AND
: 975 2076 3 .BUFFER[HM2$L_ALTIDXLBN] NEQ 0 AND
: 976 2077 3 .BUFFER[HM2$W_CLUSTER] NEQ 0 AND
: 977 2078 3 .BUFFER[HM2$W_HOMEBVN] NEQ 0 AND
: 978 2079 3 .BUFFER[HM2$W_ALHOMEBVN] NEQ 0 AND
: 979 2080 3 .BUFFER[HM2$W_ALTIDXVBN] NEQ 0 AND
: 980 2081 3 .BUFFER[HM2$W_IBMAPVBN] NEQ 0 AND
: 981 2082 3 .BUFFER[HM2$L_IBMAPLBN] NEQ 0 AND
: 982 2083 3 .BUFFER[HM2$L_MAXFILES] NEQ 0 AND
: 983 2084 3 .BUFFER[HM2$W_IBMAPSIZE] NEQ 0 AND
: 984 2085 3 .BUFFER[HM2$W_RESFILES] NEQ 0 AND
: 985 2086 3 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM2$W_CHECKSUM1)) AND
: 986 2087 3 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM2$W_CHECKSUM2))
: 987 2088 3 THEN
: 988 2089 3 BEGIN
: 989 2090 3 IF .RVN EQL 1
: 990 2091 3 THEN
: 991 2092 3 BEGIN
: 992 2093 3 FAST_STRUCLEV = 2;
: 993 2094 3 IF .BUFFER[HM2$W_SETCOUNT] GTRU MAX_VOLUMES
: 994 2095 3 THEN SIGNAL(BACKUP$ MAXVOLS, 1, INPUT_QUAL[QUAL_DEV_DESC]);
: 995 2096 3 COM_I_SETCOUNT = .BUFFER[HM2$W_SETCOUNT];
: 996 2097 3 IF .COM_I_SETCOUNT EQL 0 THEN COM_I_SETCOUNT = 1;
: 997 2098 3 IF
: 998 2099 3 .QUAL[QUAL_VOLU] AND
: 999 2100 3 .QUAL[QUAL_VOLU_VALUE] GTRU .COM_I_SETCOUNT
: 1000 2101 3 THEN
: 1001 2102 3 SIGNAL(
: 1002 2103 3 BACKUP$ NOSUCHRVN,
: 1003 2104 3 2, .QUAL[QUAL_VOLU_VALUE], .COM_I_SETCOUNT);
: 1004 2105 3 CH$MOVE(
: 1005 2106 3 HM2$S_STRUCNAME,
: 1006 2107 3 .BUFFER[HM2$T_STRUCNAME],
: 1007 2108 3 COM_I_STRUCNAME);
: 1008 2109 3 END;
: 1009 2110 3 RETURN;
: 1010 2111 3 END
: 1011 2112 3 ELSE IF
: 1012 2113 3 .RVN EQL 1 AND
: 1013 2114 3 .BUFFER[HM2$B_STRUCLEV] EQL 1 AND
: 1014 2115 3 .BUFFER[HM1$W_CLUSTER] EQL 1 AND
: 1015 2116 3 .BUFFER[HM1$L_IBMAPLBN] NEQ 0 AND
: 1016 2117 3 .BUFFER[HM1$W_MAXFILES] NEQ 0 AND
: 1017 2118 3 .BUFFER[HM1$W_IBMAPSIZE] NEQ 0 AND
: 1018 2119 3 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM1$W_CHECKSUM1)) AND
: 1019 2120 3 CHECKSUM2(.BUFFER, $BYTEOFFSET(HM1$W_CHECKSUM2))
: 1020 2121 3 THEN
: 1021 2122 3 BEGIN
: 1022 2123 3 FAST_STRUCLEV = 1;
: 1023 2124 3 COM_I_SETCOUNT = 1;
: 1024 2125 3 IF
: 1025 2126 3 .QUAL[QUAL_VOLU] AND
```



```
1026 2127 4 .QUAL[QUAL_VOLU_VALUE] GTRU 1
1027 2128 4 THEN
1028 2129 4 SIGNAL(
1029 2130 4 BACKUP$ NOSUCHRVN,
1030 2131 4 2, .QUAL[QUAL_VOLU_VALUE], .COM_I_SETCOUNT);
1031 2132 4 RETURN;
1032 2133 4 END;
1033 2134 3 END;
1034 2135 2
1035 2136 2
1036 2137 2 ! No good home block found. Report failure.
1037 2138 2
1038 2139 2 SIGNAL(BACKUP$ PROCINDEX, 2, INPUT_QUAL[QUAL_DEV_DESC], .RVN, .OLD_STATUS);
1039 2140 1 END;
```

## OFFC 00000 READ\_HOMEBLOCK:

5B	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	2014
5A	00000000'	EF	9E	00009	MOVAB	CHECKSUM2, R11	
5E		08	C2	00010	MOVAB	COM_I_SETCOUNT, R10	
59		2C	D0	00013	SUBL2	#8, SP	
56	08	AC	D0	00016	MOVL	#4, OLD STATUS	2057
57		02	D0	0001A	MOVL	BUFFER, R6	2066
		7E	7C	0001D	MOVL	#2, VBN	
		7E	D4	0001F	CLRQ	-(SP)	
		57	DD	00021	CLRL	-(SP)	
7E	0200	8F	3C	00023	PUSHL	VBN	
		56	DD	00028	MOVZWL	#512, -(SP)	
		7E	7C	0002A	PUSHL	R6	
	20	AE	9F	0002C	CLRQ	-(SP)	
	46	31	DD	0002F	PUSHAB	IOSB	
		AA	DD	00031	PUSHL	#49	
		7E	D4	00034	PUSHL	INPUT_CHAN	
00000000G	00	0C	FB	00036	CLRL	-(SP)	
58		50	D0	0003D	CALLS	#12, SYSSQIOW	
06		58	E9	00040	MOVL	R0, STATUS	
58		6E	3C	00043	BLBC	STATUS, 2\$	2067
06		58	E8	00046	MOVZWL	IOSB, STATUS	
59		58	D0	00049	BLBS	STATUS, 3\$	2069
		0111	31	0004C	MOVL	STATUS, OLD_STATUS	2071
02	0D	A6	91	0004F	BRW	12\$	
		06	12	00053	CMPB	13(R6), #2	2074
57	10	00	ED	00055	BNEQ	4\$	
		03	13	0005B	CMPZV	#0, #16, 16(R6), VBN	2075
		00A1	31	0005D	BEQL	6\$	
		08	A6	D5	BRW	11\$	
		F8	13	00063	TSTL	8(R6)	2076
		0E	A6	B5	BEQL	5\$	
		F3	13	00068	TSTW	14(R6)	2077
10		A6	B5	0006A	BEQL	5\$	
		EE	13	0006D	TSTW	16(R6)	2078
12		A6	B5	0006F	BEQL	5\$	
		E9	13	00072	TSTW	18(R6)	2079
					BEQL	5\$	

			14	A6	B5	00074	TSTW	20(R6)	2080
				E4	13	00077	BEQL	5\$	
			16	A6	B5	00079	TSTW	22(R6)	2081
				DF	13	0007C	BEQL	5\$	
			18	A6	D5	0007E	TSTL	24(R6)	2082
				7E	13	00081	BEQL	11\$	
			1C	A6	D5	00083	TSTL	28(R6)	2083
				79	13	00086	BEQL	11\$	
			20	A6	B5	00088	TSTW	32(R6)	2084
				74	13	0008B	BEQL	11\$	
			22	A6	B5	0008D	TSTW	34(R6)	2085
				6F	13	00090	BEQL	11\$	
				3A	DD	00092	PUSHL	#58	2086
				56	DD	00094	PUSHL	R6	
	6B			02	FB	00096	CALLS	#2, CHECKSUM2	
	65			50	E9	00099	BLBC	R0, 11\$	
	7E	01FE		8F	3C	0009C	MOVZWL	#510, -(SP)	2087
				56	DD	000A1	PUSHL	R6	
	6B			02	FB	000A3	CALLS	#2, CHECKSUM2	
	58			50	E9	000A6	BLBC	R0, 11\$	
	01	04		AC	D1	000A9	CMPL	RVN, #1	2090
				01	13	000AD	BEQL	7\$	
					04	000AF	RET		
	45	AA		02	90	000B0	MOVB	#2, FAST_STRUCLEV	2093
	00FF	8F	28	A6	B1	000B4	CMPL	40(R6), #255	2094
				14	1B	000BA	BLEQU	8\$	
7E	5A	AA		10	C1	000BC	ADDL3	#16, INPUT_QUAL, -(SP)	2095
				01	DD	000C1	PUSHL	#1	
		00000000G		8F	DD	000C3	PUSHL	#BACKUP\$ MAXVOLS	
	00000000G	00		03	FB	000C9	CALLS	#3, LIB\$SIGNAL	
		6A	28	A6	90	000D0	MOVB	40(R6), COM_I_SETCOUNT	2096
				03	12	000D4	BNEQ	9\$	2097
	6A			01	90	000D6	MOVB	#1, COM_I_SETCOUNT	
	1C		90	AA	E9	000D9	BLBC	QUAL+14, T0\$	2099
	6A		D1	AA	91	000DD	CMPL	QUAL+79, COM_I_SETCOUNT	2100
				16	1B	000E1	BLEQU	10\$	
	7E			6A	9A	000E3	MOVZBL	COM_I_SETCOUNT, -(SP)	2104
	7E		D1	AA	9A	000E6	MOVZBL	QUAL+79, -(SP)	
				02	DD	000EA	PUSHL	#2	2102
		00000000G		8F	DD	000EC	PUSHL	#BACKUP\$ NOSUCHRVN	
	00000000G	00		04	FB	000F2	CALLS	#4, LIB\$SIGNAL	
02	AA	01CC		0C	28	000F9	MOVCL3	#12, 460(R6), COM_I_STRUCNAME	2107
					04	00100	RET		2089
			01	AC	D1	00101	CMPL	RVN, #1	2113
				59	12	00105	BNEQ	12\$	
			01	A6	91	00107	CMPL	13(R6), #1	2114
				53	12	0010B	BNEQ	12\$	
			01	A6	B1	0010D	CMPL	8(R6), #1	2115
				4D	12	00111	BNEQ	12\$	
			02	A6	D5	00113	TSTL	2(R6)	2116
				48	13	00116	BEQL	12\$	
			06	A6	B5	00118	TSTW	6(R6)	2117
				43	13	0011B	BEQL	12\$	
				66	B5	0011D	TSTW	(R6)	2118
				3F	13	0011F	BEQL	12\$	
				3A	DD	00121	PUSHL	#58	2119
				56	DD	00123	PUSHL	R6	



FASTSCAN  
V04-000

Fast file scan  
READ\_HOMEBLOCK - read home block from index fil

K 5  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 40  
(5)

	6B		02	FB	00125	CALLS	#2, CHECKSUM2	:	
	35		50	E9	00128	BLBC	R0, 12\$	:	
	7E	01FE	8F	3C	0012B	MOVZWL	#510, -(SP)	:	2120
			56	DD	00130	PUSHL	R6	:	
	6B		02	FB	00132	CALLS	#2, CHECKSUM2	:	
	28		50	E9	00135	BLBC	R0, 12\$	:	
45	AA		01	90	00138	MOVB	#1, FAST_STRUCLEV	:	2123
	6A		01	90	0013C	MOVB	#1, COM_I_SETCOUNT	:	2124
	40	90	AA	E9	0013F	BLBC	QUAL+14, T3\$	:	2126
	01	D1	AA	91	00143	CMPB	QUAL+79, #1	:	2127
			3A	1B	00147	BLEQU	13\$	:	
	7E		6A	9A	00149	MOVZBL	COM_I_SETCOUNT, -(SP)	:	2131
	7E	D1	AA	9A	0014C	MOVZBL	QUAL+79, -(SP)	:	
			02	DD	00150	PUSHL	#2	:	2129
		00000000G	8F	DD	00152	PUSHL	#BACKUP\$ NOSUCHRVN	:	
		00	04	FB	00158	CALLS	#4, LIB\$SIGNAL	:	
				04	0015F	RET		:	2122
FEB3			8F	F1	00160	12\$:	ACBL	#100, #1, VBN, 1\$	2058
			59	DD	0016A		PUSHL	OLD_STATUS	2139
			AC	DD	0016C		PUSHL	RVN	
	7E	5A	AA	10	C1	0016F	ADDL3	#16, INPUT_QUAL, -(SP)	
			02	DD	00174		PUSHL	#2	
		00000000G	8F	DD	00176		PUSHL	#BACKUP\$ PROCINDEX	
		00	05	FB	0017C		CALLS	#5, LIB\$SIGNAL	
			04	00183	13\$:		RET		2140

; Routine Size: 388 bytes, Routine Base: CODE + 09AF

```
1041 2141 1 %SBTTL 'VERIFY_HEADER - verify file header'
1042 2142 1 ROUTINE VERIFY_HEADER(HEADER,FILE_ID)=
1043 2143 1
1044 2144 1 !++
1045 2145 1
1046 2146 1 FUNCTIONAL DESCRIPTION:
1047 2147 1 This routine determines if the block given it is a valid file header.
1048 2148 1
1049 2149 1 INPUT PARAMETERS:
1050 2150 1 HEADER - Pointer to header.
1051 2151 1 FILE_ID - Purported file ID.
1052 2152 1
1053 2153 1 IMPLICIT INPUTS:
1054 2154 1 NONE
1055 2155 1
1056 2156 1 OUTPUT PARAMETERS:
1057 2157 1 NONE
1058 2158 1
1059 2159 1 IMPLICIT OUTPUTS:
1060 2160 1 NONE
1061 2161 1
1062 2162 1 ROUTINE VALUE:
1063 2163 1 0 if invalid file header
1064 2164 1 1 if valid file header
1065 2165 1 2 if deleted file header
1066 2166 1
1067 2167 1 SIDE EFFECTS:
1068 2168 1 NONE
1069 2169 1
1070 2170 1 !--
1071 2171 1
1072 2172 2 BEGIN
1073 2173 2 MAP
1074 2174 2 HEADER: REF BBLOCK, ! file header arg
1075 2175 2 FILE_ID: REF BBLOCK; ! file ID arg
1076 2176 2
1077 2177 2
1078 2178 2 ! First check the structure level.
1079 2179 2
1080 2180 2 IF .HEADER[FH2$B_STRUCLEV] NEQ .FAST_STRUCLEV
1081 2181 2 THEN
1082 2182 2 RETURN 0;
1083 2183 2
1084 2184 2
1085 2185 2 IF .FAST_STRUCLEV EQL 2
1086 2186 2 THEN
1087 2187 2 BEGIN
1088 2188 2
1089 2189 2 ! Check the area offsets and the retrieval pointer use counts for
1090 2190 2 consistency.
1091 2191 2
1092 2192 2 IF
1093 2193 2 .HEADER[FH2$B_IDOFFSET] LSSU $BYTEOFFSET (FH2$H HIGHWATER)/2 OR
1094 2194 2 .HEADER[FH2$B_MPOFFSET] LSSU .HEADER[FH2$B_IDOFFSET] OR
1095 2195 2 .HEADER[FH2$B_ACOFFSET] LSSU .HEADER[FH2$B_MPOFFSET] OR
1096 2196 2 .HEADER[FH2$B_RSOFFSET] LSSU .HEADER[FH2$B_ACOFFSET] OR
1097 2197 2 .HEADER[FH2$B_MAP_INUSE] GTRU .HEADER[FH2$B_ACOFFSET] - .HEADER[FH2$B_MPOFFSET]
```



```
: 1098      2198      3      THEN
: 1099      2199      3      RETURN 0;
: 1100      2200      3
: 1101      2201      3
: 1102      2202      3      ! At this point, we have verified that the block at least once was a
: 1103      2203      3      ! valid file header.
: 1104      2204      3
: 1105      2205      3      ! Look at the file number in the header. If zero, this is a
: 1106      2206      3      ! deleted header.
: 1107      2207      3
: 1108      2208      3      IF
: 1109      2209      3      .HEADER[FH2$W_FID_NUM] EQL 0 AND
: 1110      2210      3      .HEADER[FH2$B_FID_NMX] EQL 0
: 1111      2211      3      THEN
: 1112      2212      3      RETURN 2;
: 1113      2213      3
: 1114      2214      3
: 1115      2215      3      ! Now compute the header checksum.
: 1116      2216      3
: 1117      2217      3      IF NOT CHECKSUM(.HEADER)
: 1118      2218      3      THEN
: 1119      2219      3      RETURN 2;
: 1120      2220      3
: 1121      2221      3
: 1122      2222      3      ! Check file number and file sequence number.
: 1123      2223      3
: 1124      2224      3      IF
: 1125      2225      3      .HEADER[FH2$W_FID_NUM] NEQ .FILE_ID[FID$W_NUM] OR
: 1126      2226      3      .HEADER[FH2$B_FID_NMX] NEQ .FILE_ID[FID$B_NMX] OR
: 1127      2227      3      .HEADER[FH2$W_FID_SEQ] NEQ .FILE_ID[FID$W_SEQ]
: 1128      2228      3      THEN
: 1129      2229      3      RETURN 2;
: 1130      2230      3      END
: 1131      2231      3      ELSE
: 1132      2232      3      BEGIN
: 1133      2233      3      LOCAL
: 1134      2234      3      MAP_AREA:      REF BBLOCK;
: 1135      2235      3
: 1136      2236      3
: 1137      2237      3      ! Now point to the map area and make sure that the extension
: 1138      2238      3      ! RVN is zero. Also check the retrieval pointer format data.
: 1139      2239      3
: 1140      2240      3      MAP_AREA = .HEADER + .HEADER[FH1$B_MPOFFSET]*2;
: 1141      2241      3      IF
: 1142      2242      3      .MAP_AREA[FM1$B_EX_RVN] NEQ 0 OR
: 1143      2243      3      .MAP_AREA[FM1$B_COUNTSIZE] NEQ 1 OR
: 1144      2244      3      .MAP_AREA[FM1$B_LBNSIZE] NEQ 3
: 1145      2245      3      THEN
: 1146      2246      3      RETURN 0;
: 1147      2247      3
: 1148      2248      3
: 1149      2249      3      ! Check the retrieval pointer counts for consistency with the
: 1150      2250      3      ! available space.
: 1151      2251      3
: 1152      2252      3      IF
: 1153      2253      3      .MAP_AREA[FM1$B_INUSE] GTRU .MAP_AREA[FM1$B_AVAIL] OR
: 1154      2254      3      .MAP_AREA[FM1$B_AVAIL] GTRU 255 = (.MAP_AREA + FM1$C_POINTERS - .HEADER) / 2
```

```
: 1155      2255 3 THEN
: 1156      2256 3 RETURN 0;
: 1157      2257 3
: 1158      2258 3
: 1159      2259 3 ! At this point, we have verified that the block at least once was a
: 1160      2260 3 ! valid file header.
: 1161      2261 3
: 1162      2262 3 ! Look at the file number in the header. If zero, this is a
: 1163      2263 3 ! deleted header.
: 1164      2264 3
: 1165      2265 3 IF .HEADER[FH1$W_FID_NUM] EQL 0
: 1166      2266 3 THEN
: 1167      2267 3 RETURN 2;
: 1168      2268 3
: 1169      2269 3
: 1170      2270 3 ! Now compute the header checksum.
: 1171      2271 3
: 1172      2272 3 IF NOT CHECKSUM(.HEADER)
: 1173      2273 3 THEN
: 1174      2274 3 RETURN 2;
: 1175      2275 3
: 1176      2276 3
: 1177      2277 3 ! Check file number and file sequence number.
: 1178      2278 3
: 1179      2279 3 IF
: 1180      2280 3 .HEADER[FH1$W_FID_NUM] NEQ .FILE_ID[FID$W_NUM] OR
: 1181      2281 3 .HEADER[FH1$W_FID_SEQ] NEQ .FILE_ID[FID$W_SEQ]
: 1182      2282 3 THEN
: 1183      2283 3 RETURN 2;
: 1184      2284 3 END;
: 1185      2285 3
: 1186      2286 3
: 1187      2287 3 ! Header is OK.
: 1188      2288 3
: 1189      2289 3 RETURN 1;
: 1190      2290 3 END;
```

```
003C 00000 VERIFY_HEADER:
55 00000000' EF 9E 00002 .WORD Save R2,R3,R4,R5 : 2142
54 00000000G 00 9E 00009 MOVAB FAST_STRUCLEV, R5
52 04 AC D0 00010 MOVAB CHECKSUM, R4
65 07 A2 91 00014 MOVL HEADER, R2 : 2180
75 12 00018 CMPB 7(R2), FAST_STRUCLEV
53 08 AC D0 0001A MOVL FILE_ID, R3 : 2225
02 65 91 0001E CMPB FAST_STRUCLEV, #2 : 2185
26 55 12 00021 BNEQ 4$
62 01 A2 91 00023 CMPB (R2), #38 : 2193
0C 1F 00026 BLSSU 1$
01 A2 02 A2 91 00028 CMPB 1(R2), (R2) : 2194
05 1F 0002E BLSSU 1$
05 1F 00033 CMPB 2(R2), 1(R2) : 2195
05 1F 00033 BLSSU 1$
```



	02	A2	03	A2	91	00035	CMPB	3(R2), 2(R2)	2196
				03	1E	0003A	BGEQU	2\$	
				0092	31	0003C	BRW	9\$	
	51		02	A2	9A	0003F	MOVZBL	2(R2), R1	2197
	50		01	A2	9A	00043	MOVZBL	1(R2), R0	
	51			50	C2	00047	SUBL2	R0, R1	
51		3A		08	00	ED	CMPZV	#0, #8, 58(R2), R1	
					7F	1A	BGTRU	9\$	
			08	A2	B5	00052	TSTW	8(R2)	2209
				05	12	00055	BNEQ	3\$	
			0D	A2	95	00057	TSTB	13(R2)	2210
				6D	13	0005A	BEQL	7\$	
				52	DD	0005C	PUSHL	R2	2217
	64			01	FB	0005E	CALLS	#1, CHECKSUM	
	65			50	E9	00061	BLBC	R0, 7\$	
	63		08	A2	B1	00064	CMPW	8(R2), (R3)	2225
				5F	12	00068	BNEQ	7\$	
05	A3		0D	A2	91	0006A	CMPB	13(R2), 5(R3)	2226
				58	12	0006F	BNEQ	7\$	
02	A3		0A	A2	B1	00071	CMPW	10(R2), 2(R3)	2227
				4F	11	00076	BRB	6\$	
	50		01	A2	9A	00078	MOVZBL	1(R2), R0	2240
	50			6240	3E	0007C	MOVAV	(R2)[R0], MAP_AREA	
			01	A0	95	00080	TSTB	1(MAP_AREA)	2242
				4C	12	00083	BNEQ	9\$	
	01		06	A0	91	00085	CMPB	6(MAP_AREA), #1	2243
				46	12	00089	BNEQ	9\$	
	03		07	A0	91	0008B	CMPB	7(MAP_AREA), #3	2244
				40	12	0008F	BNEQ	9\$	
09	A0		08	A0	91	00091	CMPB	8(MAP_AREA), 9(MAP_AREA)	2253
				39	1A	00096	BGTRU	9\$	
	52			50	C3	00098	SUBL3	MAP_AREA, R2, R1	2254
	51			0A	C2	0009C	SUBL2	#10, R1	
	51			02	C6	0009F	DIVL2	#2, R1	
	51		00FF	C1	9E	000A2	MOVAB	255(R1), R1	
51		09		08	00	ED	CMPZV	#0, #8, 9(MAP_AREA), R1	
					22	1A	BGTRU	9\$	
			02	A2	B5	000AF	TSTW	2(R2)	2265
				15	13	000B2	BEQL	7\$	
				52	DD	000B4	PUSHL	R2	2272
	64			01	FB	000B6	CALLS	#1, CHECKSUM	
	0D			50	E9	000B9	BLBC	R0, 7\$	
	63		02	A2	B1	000BC	CMPW	2(R2), (R3)	2280
				07	12	000C0	BNEQ	7\$	
02	A3		04	A2	B1	000C2	CMPW	4(R2), 2(R3)	2281
				04	13	000C7	BEQL	8\$	
	50			02	D0	000C9	MOVL	#2, R0	2283
					04	000CC	RET		
	50		01	D0	000CD		MOVL	#1, R0	2289
				04	000D0		RET		
				50	D4	000D1	CLRL	R0	2290
				04	000D3		RET		

; Routine Size: 212 bytes, Routine Base: CODE + 0B33



```
1192 2291 1 %SBTTL 'PROCESS_FILE - process selected file'
1193 2292 1 ROUTINE PROCESS_FILE: NOVALUE=
1194 2293 1
1195 2294 1 ++
1196 2295 1
1197 2296 1 FUNCTIONAL DESCRIPTION:
1198 2297 1 This routine is called when the next file that matches the selection
1199 2298 1 file specification has been located. It completes the tests that
1200 2299 1 select files to be processed, and if these are passed, processes the
1201 2300 1 file.
1202 2301 1
1203 2302 1 INPUT PARAMETERS:
1204 2303 1 NONE
1205 2304 1
1206 2305 1 IMPLICIT INPUTS:
1207 2306 1 INPUT_NAM - Contains resultant string and file ID.
1208 2307 1
1209 2308 1 OUTPUT PARAMETERS:
1210 2309 1 NONE
1211 2310 1
1212 2311 1 IMPLICIT OUTPUTS:
1213 2312 1 NONE
1214 2313 1
1215 2314 1 ROUTINE VALUE:
1216 2315 1 NONE
1217 2316 1
1218 2317 1 SIDE EFFECTS:
1219 2318 1 File processed if appropriate.
1220 2319 1
1221 2320 1 --
1222 2321 1
1223 2322 2 BEGIN
1224 2323 2 LOCAL
1225 2324 2 FILE_NUMBER, ! Clean file number
1226 2325 2 RVN; ! Clean RVN
1227 2326 2
1228 2327 2
1229 2328 2 ! Get clean file ID.
1230 2329 2
1231 2330 2 FILE_NUMBER = .INPUT_NAM[NAM$W_FID_NUM];
1232 2331 2 FILE_NUMBER<16,8> = .INPUT_NAM[NAM$B_FID_NMX];
1233 2332 2 RVN = .INPUT_NAM[NAM$B_FID_RVN];
1234 2333 2
1235 2334 2
1236 2335 2 IF .QUAL[QUAL_FAST]
1237 2336 2 THEN
1238 2337 2 BEGIN
1239 2338 2
1240 2339 2 ! First, make sure the RVN is in range. Then, make sure the file number
1241 2340 2 is in range.
1242 2341 2
1243 2342 2 IF .RVN GTRU .COM I SETCOUNT THEN RETURN;
1244 2343 2 IF .FILE_NUMBER GTRD .FAST_IMAP_SIZE[.RVN-1]*4096 THEN RETURN;
1245 2344 2
1246 2345 2
1247 2346 2 ! See if file is selected.
1248 2347 2
```



```
: 1249      2348 3      IF .QUAL[QUAL_IMAG]
: 1250      2349 3      THEN
: 1251      2350 4          BEGIN
: 1252      2351 4          IF NOT .BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1]
: 1253      2352 4          THEN
: 1254      2353 4              RETURN;
: 1255      2354 4          END
: 1256      2355 3      ELSE
: 1257      2356 4          BEGIN
: 1258      2357 4          IF
: 1259      2358 4              NOT .BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] AND
: 1260      2359 5              NOT (.DIR_STATUSED_STAT_SCANNED] AND .QUAL[QUAL_OSAV])
: 1261      2360 4          THEN
: 1262      2361 4              RETURN;
: 1263      2362 3          END;
: 1264      2363 2      END;
: 1265      2364 2
: 1266      2365 2
: 1267      2366 2      ! Finish evaluating selection criteria.
: 1268      2367 2
: 1269      2368 2      IF
: 1270      2369 2          NOT .QUAL[QUAL_IMAG] AND
: 1271      2370 3          NOT (.DIR_STATUSED_STAT_SCANNED] AND .QUAL[QUAL_OSAV])
: 1272      2371 2      THEN
: 1273      2372 2          IF NOT SELECT_INPUT_FILE(%B'010') THEN RETURN;
: 1274      2373 2
: 1275      2374 2
: 1276      2375 2      ! File is selected. Process it. If successfully processed in image mode,
: 1277      2376 2      ! clear the bitmap bit to avoid processing its synonyms.
: 1278      2377 2
: 1279      2378 2      IF SAVE_ONE_FILE()
: 1280      2379 2      THEN
: 1281      2380 2          IF .QUAL[QUAL_IMAG]
: 1282      2381 2          THEN
: 1283      2382 2              BITVECTOR[.FAST_IMAP[.RVN-1], .FILE_NUMBER-1] = FALSE;
: 1284      2383 1      END;
```

				001C 00000 PROCESS_FILE:							
				54	00000000'	EF	9E	00002	.WORD	Save R2,R3,R4	: 2292
				50	00C8	C4	D0	00009	MOVAB	QUAL+8, R4	
				52	24	A0	3C	0000E	MOVL	INPUT_NAM, R0	: 2330
52		08		10	29	A0	F0	00012	MOVZWL	36(R0), FILE_NUMBER	
				53	28	A0	9A	00018	INSV	41(R0), #16, #8, FILE_NUMBER	: 2331
				A4		06	E1	0001C	MOVZBL	40(R0), RVN	: 2332
		3D	01	08		00	ED	00021	BBC	#6, QUAL+9, 2\$	: 2335
53	76	A4				6D	1F	00027	CMPZV	#0, #8, COM_I_SETCOUNT, RVN	: 2342
						00	1F	00027	BLSSU	5\$	
				50	0234	D443	DE	00029	MOVAL	@FAST_IMAP_SIZE[RVN], R0	: 2343
		50	FC	A0		0C	78	0002F	ASHL	#12, =4(R0), R0	
				50		52	D1	00034	CMPL	FILE_NUMBER, R0	
						5D	1A	00037	BGTRU	5\$	
				50	0238	D443	DE	00039	MOVAL	@FAST_IMAP[RVN], R0	: 2351
				51	FF	A2	9E	0003F	MOVAB	-1(R2), R1	

FASTSCAN  
V04-000

Fast file scan  
PROCESS\_FILE - process selected file

E 6  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 47  
(7)

06	02	A4	03	E1	00043	BBC	#3, QUAL+10, 1\$	: 2348
11	FC	B0	51	E0	00048	BBS	R1, @-4(R0), 2\$	: 2351
				04	0004D	RET		: 2353
08	FC	B0	51	E0	0004E	BBS	R1, @-4(R0), 2\$	: 2358
3D	0297	C4	02	E1	00053	BBC	#2, DIR_STATUS, 5\$	: 2359
			07	A4	95 00059	TSTB	QUAL+15	
			38	18	0005C	BGEQ	5\$	
17	02	A4	03	E0	0005E	BBS	#3, QUAL+10, 4\$	: 2369
05	0297	C4	02	E1	00063	BBC	#2, DIR_STATUS, 3\$	: 2370
			07	A4	95 00069	TSTB	QUAL+15	
			0C	19	0006C	BLSS	4\$	
			02	DD	0006E	PUSHL	#2	: 2372
00000000G	00		01	FB	00070	CALLS	#1, SELECT_INPUT_FILE	
	1C		50	E9	00077	BLBC	R0, 5\$	
00000000G	00		00	FB	0007A	CALLS	#0, SAVE_ONE_FILE	: 2378
	12		50	E9	00081	BLBC	R0, 5\$	
0D	02	A4	03	E1	00084	BBC	#3, QUAL+10, 5\$	: 2380
	50		0238	D4	43 DE 00089	MOVAL	@FAST_IMAP(RVN), R0	: 2382
			52	D7	0008F	DECL	R2	
00	FC	B0	52	E5	00091	BBCC	R2, @-4(R0), 5\$	: 2383
			04	00096	5\$:	RET		

; Routine Size: 151 bytes, Routine Base: CODE + 0C07



```
1286 2384 1 %SBTTL 'DIR_SCAN - scan a directory'
1287 2385 1 ROUTINE DIR_SCAN(RVN): NOVALUE=
1288 2386 1
1289 2387 1 ++
1290 2388 1
1291 2389 1 FUNCTIONAL DESCRIPTION:
1292 2390 1 This routine is the driver for a directory scan.
1293 2391 1
1294 2392 1 INPUT PARAMETERS:
1295 2393 1 RVN - Relative volume number.
1296 2394 1
1297 2395 1 IMPLICIT INPUTS:
1298 2396 1 NONE
1299 2397 1
1300 2398 1 OUTPUT PARAMETERS:
1301 2399 1 NONE
1302 2400 1
1303 2401 1 IMPLICIT OUTPUTS:
1304 2402 1 NONE
1305 2403 1
1306 2404 1 ROUTINE VALUE:
1307 2405 1 NONE
1308 2406 1
1309 2407 1 SIDE EFFECTS:
1310 2408 1 NONE
1311 2409 1
1312 2410 1 --
1313 2411 1
1314 2412 2 BEGIN
1315 2413 2 FAST_RVN = .RVN;
1316 2414 2 INIT_DIR_SCAN(
1317 2415 2 .INPUT_CHAN,
1318 2416 2 .INPUT_NAM,
1319 2417 2 INPUT_QUAL[QUAL_DEV_DESC],
1320 2418 2 INPUT_QUAL[QUAL_EXP_DESC],
1321 2419 2 .QUAL[QUAL_IMAG] + .QUAL[QUAL_OSAV]^2,
1322 2420 2 .RVN,
1323 2421 2 0);
1324 2422 2 WHILE FIND_NEXT() DO PROCESS_FILE();
1325 2423 2 FREE_DIR_DATA();
1326 2424 1 END;
```

0004 00000 DIR_SCAN:										
			52	00000000'	EF	9E	00002	.WORD	Save R2	2385
			C2	04	AC	90	00009	MOVAB	INPUT_QUAL, R2	2413
					7E	D4	0000F	MOVB	RVN, FAST_RVN	2418
				04	AC	DD	00011	CLRL	-(SP)	2420
51	FF32	C2	01		03	EF	00014	PUSHL	RVN	2419
50	FF37	C2	01		07	EF	0001B	EXTZV	#3, #1, QUAL+10, R1	
					6140	DF	00022	EXTZV	#7, #1, QUAL+15, R0	
					08	C1	00025	PUSHAL	(R1)[R0]	
		7E	62		10	C1	00029	ADDL3	#8, INPUT_QUAL, -(SP)	2418
		7E	62					ADDL3	#16, INPUT_QUAL, -(SP)	2417

Fast file scan  
DIR\_SCAN - scan a directory

15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 49  
(8)

		F8	A2	DD	0002D		PUSHL	INPUT_NAM
		EC	A2	DD	00030		PUSHL	INPUT_CHAN
0000V	CF		07	FB	00033		CALLS	#7, INIT_DIR_SCAN
0000V	CF		00	FB	00038	1\$:	CALLS	#0, FIND_NEXT
	07		50	E9	0003D		BLBC	R0, 2\$
FF24	CF		00	FB	00040		CALLS	#0, PROCESS_FILE
			F1	11	00045		BRB	1\$
0000V	CF		00	FB	00047	2\$:	CALLS	#0, FREE_DIR_DATA
			04	0004C			RET	

2418  
2422  
2423  
2424

; Routine Size: 77 bytes, Routine Base: CODE + 0C9E



```
: 1328 2425 1 %SBTTL 'INIT DIR_SCAN - initialize directory scan'
: 1329 2426 1 GLOBAL ROUTINE INIT_DIR_SCAN(CHAN,NAM,DEV_DESC,SEL_DESC,FLAGS,RVN,LIMIT): NOVALUE=
: 1330 2427 1
: 1331 2428 1 !++
: 1332 2429 1
: 1333 2430 1 FUNCTIONAL DESCRIPTION:
: 1334 2431 1 This routine initializes context for a directory scan.
: 1335 2432 1
: 1336 2433 1 INPUT PARAMETERS:
: 1337 2434 1 CHANNEL - Channel assigned to volume set.
: 1338 2435 1 NAM - Pointer to name block.
: 1339 2436 1 DEV_DESC - Pointer to device name descriptor.
: 1340 2437 1 SEL_DESC - Pointer to selection filespec descriptor.
: 1341 2438 1 FLAGS - Bit 0 true to request an image scan.
: 1342 2439 1 Bit 1 true to request immediate return on terminator.
: 1343 2440 1 Bit 2 true to request return of scanned directories.
: 1344 2441 1 RVN - Relative volume number.
: 1345 2442 1 LIMIT - Pointer to vector of ODS-1 scan limits or 0.
: 1346 2443 1
: 1347 2444 1 IMPLICIT INPUTS:
: 1348 2445 1 NONE
: 1349 2446 1
: 1350 2447 1 OUTPUT PARAMETERS:
: 1351 2448 1 NONE
: 1352 2449 1
: 1353 2450 1 IMPLICIT OUTPUTS:
: 1354 2451 1 NONE
: 1355 2452 1
: 1356 2453 1 ROUTINE VALUE:
: 1357 2454 1 NONE
: 1358 2455 1
: 1359 2456 1 SIDE EFFECTS:
: 1360 2457 1 NONE
: 1361 2458 1
: 1362 2459 1 !--
: 1363 2460 1
: 1364 2461 2 BEGIN
: 1365 2462 2 MAP
: 1366 2463 2 DEV_DESC : REF VECTOR, ! Device name descriptor
: 1367 2464 2 NAM : REF BBLOCK; ! Pointer to name block
: 1368 2465 2 LOCAL
: 1369 2466 2 STATUS, ! General status value
: 1370 2467 2 LOCAL_FAB : $FAB_DECL, ! FAB for $PARSE
: 1371 2468 2 LOCAL_NAM : $NAM_DECL; ! NAM for $PARSE
: 1372 2469 2
: 1373 2470 2
: 1374 2471 2 ! Initialize the impure area.
: 1375 2472 2
: 1376 2473 2 CH$FILL(0, DIR_END-DIR_BEG, DIR_BEG);
: 1377 2474 2 DIR_FLAGS[D_INITIAL] = TRUE;
: 1378 2475 2
: 1379 2476 2
: 1380 2477 2 ! Save the parameters.
: 1381 2478 2
: 1382 2479 2 DIR_CHAN = .CHAN;
: 1383 2480 2 DIR_NAM = .NAM;
: 1384 2481 2 DIR_NAM[NAM$W_DID_NUM] = FIDSC_MFD;
```

```

: 1385 2482 2 DIR_NAM[NAM$W_DID_SEQ] = FID$C_MFD;
: 1386 2483 2 DIR_NAM[NAM$W_DID_RVN] = .RVN;
: 1387 2484 2 DIR_DEV_DESC = .DEV_DESC;
: 1388 2485 2
: 1389 2486 2 ! Determine if a rooted directory is being used. If so, do another
: 1390 2487 2 ! parse to get the root directory ID.
: 1391 2488 2
: 1392 2489 2
: 1393 2490 2 IF .NAM[NAM$V_ROOT_DIR]
: 1394 2491 2 THEN
: 1395 2492 2 BEGIN
: 1396 P 2493 2 $FAB_INIT (FAB = LOCAL_FAB,
: 1397 P 2494 2 NAM = LOCAL_NAM,
: 1398 P 2495 2 FNS = .DEV_DESC[0],
: 1399 P 2496 2 FNA = .DEV_DESC[1],
: 1400 P 2497 2 DNM = '[000000]',
: 1401 2498 2 );
: 1402 P 2499 2 $NAM_INIT (NAM = LOCAL_NAM,
: 1403 P 2500 2 NOP = NOCONCEAL,
: 1404 2501 2 );
: 1405 2502 2 STATUS = $PARSE (FAB = LOCAL_FAB);
: 1406 2503 2 IF NOT .STATUS
: 1407 2504 2 THEN SIGNAL (BACKUP$_OPENIN+ST$K_SEVERE, 1, .DEV_DESC, .STATUS, .LOCAL_FAB[FAB$$_STV]);
: 1408 2505 2
: 1409 2506 2 DIR_NAM[NAM$W_FID_NUM] = .LOCAL_NAM[NAM$W_DID_NUM];
: 1410 2507 2 DIR_NAM[NAM$W_FID_SEQ] = .LOCAL_NAM[NAM$W_DID_SEQ];
: 1411 2508 2 DIR_NAM[NAM$W_FID_RVN] = .LOCAL_NAM[NAM$W_DID_RVN];
: 1412 2509 2 IF .DIR_NAM[NAM$B_FID_RVN] EQL 0 THEN DIR_NAM[NAM$B_FID_RVN] = .RVN;
: 1413 2510 2 END
: 1414 2511 2
: 1415 2512 2
: 1416 2513 2 ! If no root directory is used, establish the MFD as the root.
: 1417 2514 2
: 1418 2515 2 ELSE
: 1419 2516 2 BEGIN
: 1420 2517 2 DIR_NAM[NAM$W_FID_NUM] = FID$C_MFD;
: 1421 2518 2 DIR_NAM[NAM$W_FID_SEQ] = FID$C_MFD;
: 1422 2519 2 DIR_NAM[NAM$W_FID_RVN] = .RVN;
: 1423 2520 2 END;
: 1424 2521 2
: 1425 2522 2 ! Initialize the result string in the name block with the name of
: 1426 2523 2 ! the MFD. Note that in the case of a rooted directory, we are lying
: 1427 2524 2 ! about its name. However, this is consistent with RMS behavior and
: 1428 2525 2 ! is generally all to the best.
: 1429 2526 2
: 1430 2527 2
: 1431 2528 2 CH$COPY (.DEV_DESC[0], .DEV_DESC[1],
: 1432 2529 2 %CHARCOUNT ('[000000]000000.DIR;1'),
: 1433 2530 2 UPLIT BYTE ('[000000]000000.DIR;1'),
: 1434 2531 2 ,
: 1435 2532 2 .NAM[NAM$B_RSS],
: 1436 2533 2 .NAM[NAM$B_RSA],
: 1437 2534 2 );
: 1438 2535 2 NAM[NAM$B_RSL] = MINU (.NAM[NAM$B_RSS], .DEV_DESC[0] + %CHARCOUNT ('[000000]000000.DIR;1'));
: 1439 2536 2 INIT_NAMEBLOCK (.NAM);
: 1440 2537 2
: 1441 2538 2 ! Initialize the level stack with the root of the selected RVN.
```



Address	Disassembly	Comment	Value
2E 30 30 30 30 30 30 5D 30 30 30 30 30 30 5B 00CEB	P.AAD:	.ASCII \[000000]\	:
5D 30 30 30 30 30 30 30 30 30 30 30 30 5B 00CF3	P.AAE:	.ASCII \[000000]000000.DIR;1\	:
31 3B 52 49 44 00D02			:
30 30 30 30 30 30 00D07	P.AAF:	.ASCII \000000\	:
	.EXTRN	SYSSPARSE	
	.ENTRY	INIT_DIR_SCAN, Save R2,R3,R4,R5,R6,R7,R8,-	: 2426
		R9,R10	:
03CC 8F 00 5A 00000000' EF 9E 00002	MOVAB	DIR_NAM, R10	:
5E FF50 CE 9E 00009	MOVAB	-176(SP), SP	:
6E FC 00 2C 0000E	MOVC5	#0, (SP), #0, #972, DIR_BEG	: 2473
			:
1A AA FC 10 88 00015	BISB2	#16, DIR_FLAGS	: 2474
AA 04 AC D0 0001B	MOVL	CHAN, DIR_CHAN	: 2479
58 08 AC D0 00020	MOVL	NAM, R8	: 2480
6A 58 D0 00024	MOVL	R8, DIR_NAM	:
56 6A D0 00027	MOVL	DIR_NAM, R6	: 2481
2A A6 00040004 8F D0 0002A	MOVL	#262148, 42(R6)	:
2E A6 18 AC B0 00032	MOVW	RVN, 46(R6)	: 2483

0050	8F	7B 00	04 35	57 AA A8 6E	0C	AC 57 05 00	DO DO E1 2C	00037 0003B 0003F 00044	MOVL MOVL BBC MOVCS	DEV_DESC, R7 R7, DIR_DEV_DESC #5, 53(R8), -2\$ #0, (SP), #0, #80, \$RMS_PTR	2484 2490 2498
			60 76 7F D8 DC E0 E4 E5	AE AE AE AD AD AD AD AD	60 5003	AE 8F 02 02 6E A7 CF 67 08 00	BO 90 90 90 9E DO 9E 90 90 2C	0004B 0004D 00053 00057 0005B 0005F 00064 0006A 0006E 00072	MOVW MOVB MOVB MOVAB MOVL MOVAB MOVB MOVB MOVCS	#20483, \$RMS_PTR #2, \$RMS_PTR+22 #2, \$RMS_PTR+31 LOCAL_NAM, \$RMS_PTR+40 4(R7), \$RMS_PTR+44 P.AAD, \$RMS_PTR+48 (R7), \$RMS_PTR+52 #8, \$RMS_PTR+53 #0, (SP), #0, #96, \$RMS_PTR	2501
0060	8F	00		6E 08 AE	6002	8F 10 AE	BO 90 9F	0007A 0007F 00083	MOVW MOVB PUSHAB	#24578, \$RMS_PTR #16, \$RMS_PTR+8 LOCAL_FAB	2502
			00000000G	00 16	60	01 50 AE	FB E8 DD	00086 0008D 00090	CALLS BLBS PUSHL	#1, SYSSPARSE STATUS, 1\$ LOCAL_FAB+12	2503 2504
					6C	50 57 01	DD DD DD	00093 00095 00097	PUSHL PUSHL PUSHL	STATUS R7 #1	
			00000000G	00	00000000G	8F 05 6A	DD FB DO	00099 0009F 000A6	PUSHL CALLS MOVL	#BACKUP\$, OPENIN+4 #5, LIB\$SIGNAL DIR_NAM, R0	2506
			24 28	A0 A0	2A 2E 28	AE AE A0	DO BO 95	000A9 000AE 000B3	MOVL MOVW TSTB	LOCAL_NAM+42, 36(R0) LOCAL_NAM+46, 40(R0) 40(R0)	2508 2509
			28	A0	18	AC 0D 8F	90 11 DO	000B6 000B8 000BD	BNEQ MOVB BRB	3\$ RVN, 40(R0) 3\$	2490
			24 28	A6 A6 59	00040004	0D 8F AC	11 DO BO	000BD 000BF 000C7	2\$: MOVL MOVW	#262148, 36(R6) RVN, 40(R6)	2517 2519
			59	56 B7	04	A8 A8 67	9A DO 2C	000CC 000D0 000D4	3\$: MOVZBL MOVL MOVCS	2(R8), R9 4(R8), R6 (R7), @4(R7), #32, R9, (R6)	2532 2533
						66 0E 67	18 00 C0	000DA 000DB 000DD	BGEQ ADDL2	4\$ (R7), R6	
				56 59		67 67	00 C2	000E0 000E3	SUBL2	(R7), R9	
59	20	04	FEFE	CF		14 66	2C	000E3 000EA	MOVCS	#20, P.AAE, #32, R9, (R6)	
			51	67 50 51		14 A8 50	C1 9A D1	000EB 000EF 000F3	4\$: ADDL3 MOVZBL CMPL	#20, (R7), R1 2(R8), R0 R0, R1	2535
				50 A8		03 51 50	1B DO 90	000F6 000F8 000FB	BLEQU MOVL MOVB	5\$ R1, R0 R0, 3(R8)	
			03	50 A8		58 01	DD FB	000FF 00101	PUSHL CALLS	R8 #1, INIT_NAMEBLOCK	2536
			00000000G	00	015C	01 CA	9E	0010F 00113	MOVAB MOVB	DIR_STACK, DIR_SP #1, DIR_LEVELS	2540 2541
			03C0 19	CA AA 50		6A A0	DO	00113 00116	MOVL	DIR_NAM, R0	2542
			0178 017C	CA CA	24 28	A0 A0	DO BO	00116 0011C	MOVL MOVW	36(R0), DIR_STACK+28 40(R0), DIR_STACK+32	2544



FASTSCAN  
V04-000

Fast file scan  
INIT\_DIR\_SCAN - initialize directory scan

L 6  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 54  
(9)

		0170	CA		01	D0	00122	MOVL	#1, DIR_STACK+20	:	2545
			7E	10	AC	7D	00127	MOVQ	SEL_DEST, -(SP)	:	2550
		0000V	CF		02	FB	00128	CALLS	#2, RESET_DIR_SPEC	:	
				1C	AC	D5	00130	TSTL	LIMIT	:	2551
					07	13	00133	BEQL	6\$	:	
	03C8	CA	1C	BC	24	28	00135	MOVCL	#36, @LIMIT, DIR_SCANLIMIT	:	
		14	35	A8	05	E1	0013C	BBC	#5, 53(R8), 8\$	:	2559
08	AA	20	FEB4	CF	06	2D	00141	CMPC5	#6, P.AAF, #32, DIR_SEL_DIR, @DIR_SEL_DIR+4	:	2563
					0C	BA	00149			:	
					04	12	00148	BNEQ	7\$	:	
		1A	AA		20	88	0014D	BISB2	#32, DIR_FLAGS	:	2568
		1A	AA		10	8A	00151	BICB2	#16, DIR_FLAGS	:	2569
					04	00155	8\$:	RET		:	2572

; Routine Size: 342 bytes, Routine Base: CODE + 0D0D

```
: 1477 2573 1 %SBTTL 'RESET DIR SPEC - reinitialize directory context'
: 1478 2574 1 GLOBAL ROUTINE RESET_DIR_SPEC(SEL_DESC,FLAGS): NOVALUE=
: 1479 2575 1
: 1480 2576 1 !++
: 1481 2577 1
: 1482 2578 1 FUNCTIONAL DESCRIPTION:
: 1483 2579 1 This routine reinitializes context to change the selection file spec.
: 1484 2580 1
: 1485 2581 1 INPUT PARAMETERS:
: 1486 2582 1 SEL_DESC - Pointer to selection filespec descriptor.
: 1487 2583 1 FLAGS - Bit 0 true to request an image scan.
: 1488 2584 1 Bit 1 true to request immediate return on terminator.
: 1489 2585 1 Bit 2 true to request return of scanned directories.
: 1490 2586 1
: 1491 2587 1 IMPLICIT INPUTS:
: 1492 2588 1 NONE
: 1493 2589 1
: 1494 2590 1 OUTPUT PARAMETERS:
: 1495 2591 1 NONE
: 1496 2592 1
: 1497 2593 1 IMPLICIT OUTPUTS:
: 1498 2594 1 NONE
: 1499 2595 1
: 1500 2596 1 ROUTINE VALUE:
: 1501 2597 1 NONE
: 1502 2598 1
: 1503 2599 1 SIDE EFFECTS:
: 1504 2600 1 NONE
: 1505 2601 1
: 1506 2602 1 !--
: 1507 2603 1
: 1508 2604 2 BEGIN
: 1509 2605 2 INIT_SEL_INFO(.SEL_DESC, DIR_SEL_DIR, DIR_SEL_NTV, DIR_SEL_LATEST);
: 1510 2606 2 DIR_FLAGS[D_IMAGE_SCAN] = .FLAGS<0,1>;
: 1511 2607 2 DIR_FLAGS[D_HARD_STOP] = .FLAGS<1,1>;
: 1512 2608 2
: 1513 2609 2 DIR_FLAGS[D_SCANNED_DIRS] = FALSE;
: 1514 2610 2 IF
: 1515 2611 2 NOT .QUAL[QUAL INTE] AND
: 1516 2612 2 .DIR_SEL_NTV[DSCSW_LENGTH] EQL 5 AND
: 1517 2613 2 CH$EQL(5, .DIR_SEL_NTV[DSCSA_POINTER], 5, UPLIT BYTE ('*.*;*))
: 1518 2614 2 THEN
: 1519 2615 2 DIR_FLAGS[D_SCANNED_DIRS] = .FLAGS<2,1>;
: 1520 2616 2
: 1521 2617 2 IF .DIR_FLAGS[D_IMAGE_SCAN] THEN DIR_SEL_LATEST = +1;
: 1522 2618 2 CH$FILL(1, D_K_NLEVECS*%UPVAL, DIR_SCANLIMIT);
: 1523 2619 2
: 1524 2620 2 INCRA D FROM DIR_STACK TO .DIR_SP BY D_S_ENTRY DO
: 1525 2621 2 BEGIN
: 1526 2622 2 MAP
: 1527 2623 2
: 1528 2624 2 D: REF BBLOCK; ! Pointer to stack entry
: 1529 2625 2 LOCAL
: 1530 2626 2 STATUS, ! Status return
: 1531 2627 2 DESC: VECTOR[2]; ! Descriptor for directory string
: 1532 2628 2
: 1533 2629 2
```



```
: 1534      2630      3      ! Establish the descriptor for the directory string at the current level.
: 1535      2631      3      !
: 1536      2632      3      IF .D EQLA DIR_STACK
: 1537      2633      3      THEN
: 1538      2634      4          BEGIN
: 1539      2635      4              DESC[0] = %CHARCOUNT('000000');
: 1540      2636      4              DESC[1] = UPLIT BYTE('000000');
: 1541      2637      4          END
: 1542      2638      3      ELSE
: 1543      2639      4          BEGIN
: 1544      2640      4              IF .D EQLA .DIR_SP
: 1545      2641      4                  THEN DESC[0] = .DIR_STRING[0]
: 1546      2642      4                  ELSE DESC[0] = .BBLOCK[D + D_S_ENTRY, D_SAV_LEN];
: 1547      2643      4              DESC[1] = DIR_STRING[1];
: 1548      2644      4          END;
: 1549      2645      4
: 1550      2646      4
: 1551      2647      4      ! Allocate the dynamic areas for this level.
: 1552      2648      4      !
: 1553      2649      4      BBLOCK[D[D_TERM_DESC], DSC$A_POINTER] = GET_VM(DIR$$NAME);
: 1554      2650      4      BBLOCK[D[D_NAME_DESC], DSC$A_POINTER] = GET_VM(DIR$$NAME);
: 1555      2651      4
: 1556      2652      4
: 1557      2653      4      ! Establish the match bit and the terminator specification for this level.
: 1558      2654      4      !
: 1559      2655      4      STATUS = MATCH_DIRECTORY(
: 1560      2656      4          DESC,
: 1561      2657      4          DIR_SEL DIR,
: 1562      2658      4          D[D_TERM_DESC],
: 1563      2659      4          D[D_TERM_VER],
: 1564      2660      4          DIR_SEL NTV);
: 1565      2661      4      IF .STATUS<0,1> THEN D[D_DIR_MATCHES] = TRUE;
: 1566      2662      4      IF .STATUS<2,1> THEN D[D_WILD_TERM] = TRUE;
: 1567      2663      4      IF .STATUS<3,1> THEN D[D_NON_TERM] = TRUE;
: 1568      2664      4
: 1569      2665      4      END;
: 1570      2666      1      END;
```

```
30 2A 3B 2A 2E 2A 00E63 P.AAG: .ASCII \*.*:\
30 30 30 30 30 30 00E68 P.AAH: .ASCII \000000\
```

						00FC 0000	.ENTRY	RESET DIR SPEC, Save R2,R3,R4,R5,R6,R7	: 2574
		57	00000000G	00	9E	00002	MOVAB	GET_VM, R7	:
		56	00000000'	EF	9E	00009	MOVAB	DIR_FLAGS, R6	:
		5E		08	C2	00010	SUBL2	#8, SP	:
			03AA	C6	9F	00013	PUSHAB	DIR_SEL_LATEST	: 2605
			F6	A6	9F	00017	PUSHAB	DIR_SEL_NTV	:
			EE	A6	9F	0001A	PUSHAB	DIR_SEL_DIR	:
			04	AC	DD	0001D	PUSHL	SEL_DESC	:
			00000000G	00	04	FB	CALLS	#4, INIT_SEL_INFO	:
66		01		00	08	AC	INSV	FLAGS, #0, #T, DIR_FLAGS	: 2606
50	08	AC		01	01	EF	EXTZV	#1, #1, FLAGS, R0	: 2607
66		01		02	50	F0	INSV	R0, #2, #1, DIR_FLAGS	:



			66		08	8A	00038	BICB2	#8, DIR_FLAGS	2609
	19	FD70	C6		01	E0	0003B	BBS	#1, QUAC+14, 1\$	2611
			05	F6	A6	B1	00041	CMPW	DIR_SEL_NTV, #5	2612
					13	12	00045	BNEQ	1\$	
	A8	AF	FA	B6	05	29	00047	CMPC3	#5, @DIR_SEL_NTV+4, P.AAG	2613
					0B	12	0004D	BNEQ	1\$	
50	08	AC	01		02	EF	0004F	EXTZV	#2, #1, FLAGS, R0	2615
66		01			50	F0	00055	INSV	R0, #3, #1, DIR_FLAGS	
			05		66	E9	0005A	BLBC	DIR_FLAGS, 2\$	2617
			C6	03AA	01	D0	0005D	MOVL	#1, DIR_SEL_LATEST	
24	FF	8F	6E		00	2C	00062	MOVCS	#0, (SPT, #=1, #36, DIR_SCANLIMIT	2618
				03AE	C6		00068			
			52	0142	C6	9E	0006B	MOVAB	DIR_STACK, R2	2621
			53	03A6	C6	D0	00070	MOVL	DIR_SP, R3	
					72	11	00075	BRB	11\$	
			50	0142	C6	9E	00077	MOVAB	DIR_STACK, R0	2632
			50		52	D1	0007C	CMPL	D, R0	
					0B	12	0007F	BNEQ	4\$	
			6E		06	D0	00081	MOVL	#6, DESC	2635
	04	AE		FF72	CF	9E	00084	MOVAB	P.AAH, DESC+4	2636
					16	11	0008A	BRB	7\$	2632
	03A6	C6			52	D1	0008C	CMPL	D, DIR_SP	2640
			6E	02	06	12	00091	BNEQ	5\$	
					A6	9A	00093	MOVZBL	DIR_STRING, DESC	2641
			6E	66	04	11	00097	BRB	6\$	
					A2	9A	00099	MOVZBL	102(D), DESC	2642
	04	AE		03	A6	9E	0009D	MOVAB	DIR_STRING+1, DESC+4	2643
		7E		50	8F	9A	000A2	MOVZBL	#80, -(SP)	2649
		67			01	FB	000A6	CALLS	#1, GET_VM	
	28	A2			50	D0	000A9	MOVL	R0, 40(D)	
		7E		50	8F	9A	000AD	MOVZBL	#80, -(SP)	2650
		67			01	FB	000B1	CALLS	#1, GET_VM	
	30	A2			50	D0	000B4	MOVL	R0, 48(D)	
				F6	A6	9F	000B8	PUSHAB	DIR_SEL_NTV	2659
				38	A2	9F	000BB	PUSHAB	56(D)	
				24	A2	9F	000BE	PUSHAB	36(R2)	2658
				EE	A6	9F	000C1	PUSHAB	DIR_SEL_DIR	2655
				10	AE	9F	000C4	PUSHAB	DESC	
	00000000G	00			05	FB	000C7	CALLS	#5, MATCH_DIRECTORY	2659
		04			50	E9	000CE	BLBC	STATUS, 8\$	2661
	04	23	A2		02	88	000D1	BISB2	#2, 35(D)	
			50		02	E1	000D5	BBC	#2, STATUS, 9\$	2662
	04	23	A2		04	88	000D9	BISB2	#4, 35(D)	
			50		03	E1	000DD	BBC	#3, STATUS, 10\$	2663
	04	23	A2		08	88	000E1	BISB2	#8, 35(D)	
			52	44	A2	9E	000E5	MOVAB	68(R2), D	2621
			53		52	D1	000E9	CMPL	D, R3	
					89	1B	000EC	BLEQU	3\$	
					04	00	000EE	RET		2666

; Routine Size: 239 bytes, Routine Base: CODE + 0E6E



```
: 1572 2667 1 %SBTTL 'FIND NEXT - find next file'
: 1573 2668 1 GLOBAL ROUTINE FIND_NEXT=
: 1574 2669 1
: 1575 2670 1 ++
: 1576 2671 1
: 1577 2672 1 FUNCTIONAL DESCRIPTION:
: 1578 2673 1 This routine searches for the next file matching the specified
: 1579 2674 1 selection filespec.
: 1580 2675 1
: 1581 2676 1 INPUT PARAMETERS:
: 1582 2677 1 NONE
: 1583 2678 1
: 1584 2679 1 IMPLICIT INPUTS:
: 1585 2680 1 Directory scan context.
: 1586 2681 1
: 1587 2682 1 OUTPUT PARAMETERS:
: 1588 2683 1 NONE
: 1589 2684 1
: 1590 2685 1 IMPLICIT OUTPUTS:
: 1591 2686 1 Directory scan context. If a file was found, the name block contains
: 1592 2687 1 the resultant string and file ID.
: 1593 2688 1
: 1594 2689 1 ROUTINE VALUE:
: 1595 2690 1 True if a file was found, false otherwise.
: 1596 2691 1
: 1597 2692 1 SIDE EFFECTS:
: 1598 2693 1 NONE
: 1599 2694 1
: 1600 2695 1 --
: 1601 2696 1
: 1602 2697 2 BEGIN
: 1603 2698 2 LOCAL
: 1604 2699 2 N: REF BBLOCK, ! Local copy of DIR_NAM
: 1605 2700 2 D: REF BBLOCK; ! Local copy of DIR_SP
: 1606 2701 2
: 1607 2702 2
: 1608 2703 2 N = .DIR_NAM;
: 1609 2704 2 D = .DIR_SP;
: 1610 2705 2 DIR_STATOS = DIR_VERLIMIT = 0;
: 1611 2706 2
: 1612 2707 2
: 1613 2708 2 ! If the directory stack is now empty, we have completed the MFD.
: 1614 2709 2
: 1615 2710 2 IF .DIR_LEVELS LEQ 0
: 1616 2711 2 THEN
: 1617 2712 3 BEGIN
: 1618 2713 3 N[NAM$B_RSL] = 0;
: 1619 2714 3 N[NAM$W_FID_NUM] = N[NAM$W_FID_SEQ] = N[NAM$W_FID_RVN] = 0;
: 1620 2715 3 N[NAM$W_DID_NUM] = N[NAM$W_DID_SEQ] = N[NAM$W_DID_RVN] = 0;
: 1621 2716 3 RETURN FALSE;
: 1622 2717 3 END;
: 1623 2718 2
: 1624 2719 2
: 1625 2720 2 ! Return the MFD if requested.
: 1626 2721 2
: 1627 2722 2 IF TESTBITSC(DIR_FLAGS[D_INITIAL])
: 1628 2723 2 THEN
```

```
: 1629      2724 2      IF .DIR_FLAGS[D_SCANNED_DIRS] AND .D[D_NON_TERM]
: 1630      2725 2      THEN
: 1631      2726 2          BEGIN
: 1632      2727 2          DIR_STATUS[D_STAT_VALID] = DIR_STATUS[D_STAT_SCANNED] = TRUE;
: 1633      2728 2          RETURN TRUE;
: 1634      2729 2          END;
: 1635      2730 2
: 1636      2731 2
: 1637      2732 2      ! Loop until we find something or traverse past a possible match.
: 1638      2733 2
: 1639      2734 2      WHILE TRUE DO
: 1640      2735 2          BEGIN
: 1641      2736 2          LOCAL
: 1642      2737 2              R:          REF BBLOCK,      ! Local copy of D_REC
: 1643      2738 2              V:          REF BBLOCK;      ! Local copy of D_VER
: 1644      2739 2
: 1645      2740 2
: 1646      2741 2          ! Push down to a lower directory if necessary. This logic depends upon the
: 1647      2742 2          ! resultant string in the name block being unmodified from the previous
: 1648      2743 2          ! call.
: 1649      2744 2
: 1650      2745 2          IF NOT (.DIR_FLAGS[D_SCAN_FAILED] OR .DIR_FLAGS[D_ROOT_MFD])
: 1651      2746 2          THEN IF TESTBITSC(D[D_DIR_SCAN])
: 1652      2747 2              THEN
: 1653      2748 2                  BEGIN
: 1654      2749 2                      ! Push down the directory stack.
: 1655      2750 2                      !
: 1656      2751 2                      DIR_LEVELS = .DIR_LEVELS + 1;
: 1657      2752 2                      DIR_SP = D = .D + D_S_ENTRY;
: 1658      2753 2                      BBLOCK[D[D_FID], FID$W_NUM] = .N[NAM$W_FID_NUM];
: 1659      2754 2                      BBLOCK[D[D_FID], FID$W_SEQ] = .N[NAM$W_FID_SEQ];
: 1660      2755 2                      BBLOCK[D[D_FID], FID$W_RVN] = .N[NAM$W_FID_RVN];
: 1661      2756 2                      D[D_SAV_LEN] = .DIR_STRING[0];
: 1662      2757 2                      D[D_VBN] = 1;
: 1663      2758 2
: 1664      2759 2
: 1665      2760 2
: 1666      2761 2          ! Generate the new directory string. If this is not the top level,
: 1667      2762 2          ! append a dot, and then append the directory name.
: 1668      2763 2          !
: 1669      2764 2          IF .DIR_STRING[0] NEQ 0 THEN DIR_STRING[0] = .DIR_STRING[0] + 1;
: 1670      2765 2          DIR_STRING[0] = .DIR_STRING[0] + .N[NAM$B_NAME];
: 1671      2766 2          END;
: 1672      2767 2
: 1673      2768 2
: 1674      2769 2      ! Get a new chunk of directory if necessary.
: 1675      2770 2
: 1676      2771 2      IF
: 1677      2772 2          .D[D_REC] GEQA .D[D_BUF_LIM] AND
: 1678      2773 2          (.D[D_VBN] EQL 1 OR .D[D_VBN] LEQU .D[D_DIR_LEN])
: 1679      2774 2      THEN
: 1680      2775 2          BEGIN
: 1681      2776 2          LITERAL
: 1682      2777 2              ATR_LENGTH= MAXU(
: 1683      2778 2                  $BYTEOFFSET(FH1$W_RECATTR)+32,
: 1684      2779 2                  $BYTEOFFSET(FH2$W_FILEPROT)+2);
: 1685      2780 2          LOCAL
```



```
1686      2781  4      FIB:      BBLOCK[FIB$C_LENGTH],      ! FIB
1687      2782  4      FIB_DESC:  VECTOR[2],              ! Descriptor for FIB
1688      2783  4      STATUS,      ! Status variable
1689      2784  4      IOSB:      VECTOR[4,WORD],          ! I/O status block
1690      2785  4      HEADER:    BBLOCK[ATR_LENGTH],      ! Beginning of file header
1691      2786  4      ATR_DESC:   BBLOCK[12];              ! ACP attributes list
1692      2787  4
1693      2788  4
1694      2789  4      ! Access the current directory file.
1695      2790  4
1696      2791  4      FIB_DESC[0] = FIB$C_LENGTH;
1697      2792  4      FIB_DESC[1] = FIB;
1698      2793  4      CH$FILL (0, FIB$C_LENGTH, FIB);
1699      2794  4      FIB[FIB$L_ACCTL] = FIB$M_NORECORD;
1700      2795  4      IF .QUAL[QUAL_IGNORE_INT] THEN FIB[FIB$L_ACCTL] = FIB$M_NOLOCK OR FIB$M_NORECORD;
1701      2796  4      FIB[FIB$W_FID_NUM] = .BBLOCK[D[D_FID], FID$W_NUM];
1702      2797  4      FIB[FIB$W_FID_SEQ] = .BBLOCK[D[D_FID], FID$W_SEQ];
1703      2798  4      FIB[FIB$W_FID_RVN] = .BBLOCK[D[D_FID], FID$W_RVN];
1704      2799  4      ATR_DESC[0,0,16,0] = ATR_LENGTH;
1705      2800  4      ATR_DESC[2,0,16,0] = ATR$C_HEADER;
1706      2801  4      ATR_DESC[4,0,32,0] = HEADER;
1707      2802  4      ATR_DESC[8,0,32,0] = 0;
1708      2803  4      STATUS = C$QIOW(
1709      2804  4          FUNC=IOS_ACCESS OR IOS$M_ACCESS,
1710      2805  4          CHAN=.DIR_CHAN,
1711      2806  4          IOSB=IOSB,
1712      2807  4          P1=FIB_DESC,
1713      2808  4          P5=ATR_DESC);
1714      2809  4      IF .STATUS THEN STATUS = .IOSB[0];
1715      2810  4
1716      2811  4
1717      2812  4      ! If a privilege violation occurred and the file specification is
1718      2813  4      ! nonwild, try an ACP call in case the problem is an execute-only
1719      2814  4      ! directory.
1720      2815  4
1721      2816  4      IF
1722      2817  4          .STATUS<0,16> EQL SS$ NOPRIV AND
1723      2818  4          NOT .COM_FLAGS[COM_STANDALONE] AND
1724      2819  4          NOT .D[D_WILD_TERM] AND
1725      2820  4          .D[D_BUF_ADDR] EQL 0
1726      2821  4      THEN
1727      2822  5          BEGIN
1728      2823  5              LOCAL
1729      2824  5                  FIB_DESC:      VECTOR[2],              ! Descriptor for FIB
1730      2825  5                  FIB:      BBLOCK[FIB$C_LENGTH],      ! FIB
1731      2826  5                  FNA_DESC:    VECTOR[2],              ! Descriptor for FNA
1732      2827  5                  FNA:      VECTOR[86,BYTE];          ! Buffer for 'n.t.v'
1733      2828  5
1734      2829  5
1735      2830  5          D[D_BUF_LEN] = 512;
1736      2831  5          D[D_BUF_ADDR] = GET_VM(512);
1737      2832  5
1738      2833  5
1739      2834  5          ! Initialize the FIB.
1740      2835  5
1741      2836  5          FIB_DESC[0] = FIB$C_LENGTH;
1742      2837  5          FIB_DESC[1] = FIB;
```

```
: 1743      2838 5      CH$FILL(0, FIB$C_LENGTH, FIB);
: 1744      2839 5      FIB[FIB$W_DID_NUM] = .BBLOCK[D[D_FID], FID$W_NUM];
: 1745      2840 5      FIB[FIB$W_DID_SEQ] = .BBLOCK[D[D_FID], FID$W_SEQ];
: 1746      2841 5      FIB[FIB$W_DID_RVN] = .BBLOCK[D[D_FID], FID$W_RVN];
: 1747      2842 5
: 1748      2843 5
: 1749      2844 5      ! Initialize the filename.
: 1750      2845 5      !
: 1751      2846 5      FNA_DESC[0] = %ALLOCATION(FNA);
: 1752      2847 5      FNA_DESC[1] = FNA;
: 1753      2848 5      $FAD(
: 1754      2849 5      $DESCRIPTOR('!AS;!UW'),
: 1755      2850 5      FNA_DESC,
: 1756      2851 5      FNA_DESC,
: 1757      2852 5      D[D_TERM_DESC], .D[D_TERM_VER]);
: 1758      2853 5
: 1759      2854 5
: 1760      2855 5      ! Execute the lookup.
: 1761      2856 5      !
: 1762      2857 5      STATUS = $QIOW(
: 1763      2858 5      FUNC=IOS$ ACCESS,
: 1764      2859 5      CHAN=.DIR_CHAN,
: 1765      2860 5      IOSB=IOSB,
: 1766      2861 5      P1=FIB_DESC,
: 1767      2862 5      P2=FNA_DESC);
: 1768      2863 5      IF .STATUS THEN STATUS = .IOSB[0];
: 1769      2864 5
: 1770      2865 5
: 1771      2866 5      ! If a no such file error occurred, simulate an empty directory.
: 1772      2867 5      ! Otherwise, simulate a single block directory that contains the
: 1773      2868 5      ! desired entry.
: 1774      2869 5
: 1775      2870 5      IF .STATUS<0,16> EQL SS$_NOSUCHFILE
: 1776      2871 5      THEN
: 1777      2872 6      BEGIN
: 1778      2873 6      D[D_DIR_LEN] = 0;
: 1779      2874 6      STATUS = SS$_NORMAL;
: 1780      2875 6      END
: 1781      2876 5      ELSE
: 1782      2877 6      BEGIN
: 1783      2878 6      LOCAL
: 1784      2879 6      L,
: 1785      2880 6      P:
: 1786      2881 6      REF BBLOCK;
: 1787      2882 6      ! Length of name and type
: 1788      2883 6      ! Pointer to entry
: 1789      2884 6
: 1790      2885 6      ! Initialize directory stack context.
: 1791      2886 6      !
: 1792      2887 6      DIR_STRUCLEV = 2;
: 1793      2888 6      D[D_VBN] = 2;
: 1794      2889 6      D[D_DIR_LEN] = 1;
: 1795      2890 6
: 1796      2891 6      ! Initialize simulated directory entry.
: 1797      2892 6      !
: 1798      2893 6      L = (.BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH] + 1) AND NOT 1;
: 1799      2894 6      P = .D[D_BUF_ADDR];
: 1799      2894 6      P[DIR$W_SIZE] = .L + DIR$C_LENGTH - 2 + DIR$C_VERSION;
```



```
: 1800      2895  6      P[DIR$W_VERLIMIT] = .FIB[FIB$W_VERLIMIT];
: 1801      2896  6      P[DIR$B_FLAGS] = 0;
: 1802      2897  6      P[DIR$B_NAMECOUNT] = .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH];
: 1803      2898  6      P = CH$COPY(
: 1804      2899  6          .BBLOCK[D[D_TERM_DESC], DSC$W_LENGTH],
: 1805      2900  6          .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER],
: 1806      2901  6          0,
: 1807      2902  6          .L, P[DIR$T_NAME]);
: 1808      2903  6      P[DIR$W_VERSION] = .D[D_TERM_VER];
: 1809      2904  6      P[DIR$W_FID_NUM] = .FIB[FIB$W_FID_NUM];
: 1810      2905  6      P[DIR$W_FID_SEQ] = .FIB[FIB$W_FID_SEQ];
: 1811      2906  6      P[DIR$W_FID_RVN] = .FIB[FIB$W_FID_RVN];
: 1812      2907  6
: 1813      2908  6
: 1814      2909  6      D[D_BUF_LIM] = .P + DIR$C_VERSION;
: 1815      2910  5      END;
: 1816      2911  4      END;
: 1817      2912  4
: 1818      2913  4
: 1819      2914  4      IF NOT .STATUS
: 1820      2915  4      THEN
: 1821      2916  5          BEGIN
: 1822      2917  5              ! Report failure to access the directory.
: 1823      2918  5              !
: 1824      2919  5              SIGNAL(
: 1825      2920  5                  BACKUP$_OPENDIR,
: 1826      2921  5                  2,
: 1827      2922  5                  .DIR_DEV_DESC,
: 1828      2923  5                  (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
: 1829      2924  5                  .STATUS);
: 1830      2925  5
: 1831      2926  5              !
: 1832      2927  5              ! Readjust context so that processing of the directory is
: 1833      2928  5              ! aborted.
: 1834      2929  5              !
: 1835      2930  5              D[D_DIR_LEN] = 0;
: 1836      2931  5              END
: 1837      2932  5      ELSE
: 1838      2933  4          BEGIN
: 1839      2934  5              !
: 1840      2935  5              ! If there is not currently a buffer, this is the first chunk of
: 1841      2936  5              ! this directory. Determine whether the file is indeed a
: 1842      2937  5              ! directory, determine its size, and allocate the buffer.
: 1843      2938  5              !
: 1844      2939  5              !
: 1845      2940  5              IF .D[D_BUF_ADDR] EQL 0
: 1846      2941  5              THEN
: 1847      2942  6                  BEGIN
: 1848      2943  6
: 1849      2944  6                  ! Ensure that the file is, in fact, a directory. At this point
: 1850      2945  6                  ! we know only that the filename is ".DIR;1". Use the portion
: 1851      2946  6                  ! of the file header that was obtained during the access.
: 1852      2947  6                  ! Compute the file length if valid. If invalid, leave the file
: 1853      2948  6                  ! length zero to avoid processing the file.
: 1854      2949  6                  !
: 1855      2950  6                  D[D_DIR_LEN] = 0;
: 1856      2951  6                  DIR_STRUCLEV = .HEADER[FH2$B_STRUCLEV];
```

```
1857 2952 6 IF .DIR_STRUCLEV EQL 2
1858 2953 6 THEN
1859 2954 7 BEGIN
1860 2955 7 BIND
1861 2956 7 RECATTR= HEADER[FH2$W_RECATTR]: BBLOCK;
1862 2957 7
1863 2958 7 IF .HEADER[FH2$V_DIRECTORY]
1864 2959 7 THEN
1865 2960 8 BEGIN
1866 2961 8 D[D_DIR_LEN] = ROT(.RECATTR[FAT$L_EFBLK], 16);
1867 2962 8 IF .RECATTR[FAT$W_FFBYTE] EQL 0
1868 2963 8 THEN D[D_DIR_LEN] = .D[D_DIR_LEN] - 1;
1869 2964 8 D[D_FPRO] = .HEADER[FH2$W_FICEPROT];
1870 2965 8 D[D_UIC] = .HEADER[FH2$L_FILEOWNER];
1871 2966 8 D[D_VERLIM] = .BBLOCK[HEADER[FH2$W_RECATTR], FAT$W_VERSIONS];
1872 2967 7 END;
1873 2968 7
1874 2969 6 ELSE
1875 2970 7 BEGIN
1876 2971 7 BIND
1877 2972 7 RECATTR= HEADER[FH1$W_RECATTR]: BBLOCK;
1878 2973 7
1879 2974 7 IF
1880 2975 7 .RECATTR[FAT$B_RTYPE] EQL FAT$C_FIXED AND
1881 2976 7 .RECATTR[FAT$W_RSIZ] EQL NMB$C_DIRENTRY
1882 2977 7 THEN
1883 2978 8 BEGIN
1884 2979 8 D[D_DIR_LEN] = ROT(.RECATTR[FAT$L_EFBLK], 16);
1885 2980 8 IF .RECATTR[FAT$W_FFBYTE] EQL 0
1886 2981 8 THEN D[D_DIR_LEN] = .D[D_DIR_LEN] - 1;
1887 2982 8 D[D_FPRO] = .HEADER[FH1$W_FICEPROT];
1888 2983 8 D[D_UIC] = .HEADER[FH1$B_OICMEMBER];
1889 2984 8 (D[D_UIC])<16,16> = .HEADER[FH1$B_UICGROUP];
1890 2985 8 D[D_VERLIM] = .BBLOCK[HEADER[FH1$W_RECATTR], FAT$W_VERSIONS];
1891 2986 7 END;
1892 2987 6
1893 2988 6
1894 2989 6
1895 2990 6 ! If the file looks like a directory and is not zero length,
1896 2991 6 ! allocate a buffer for it.
1897 2992 6
1898 2993 6 IF .D[D_DIR_LEN] NEQ 0
1899 2994 6 THEN
1900 2995 7 BEGIN
1901 2996 7
1902 2997 7 ! Compute buffer length. Try to read the entire directory
1903 2998 7 ! at one time, but no more than DIR_BUF_COUNT blocks.
1904 2999 7 ! However, for an ODS-1 directory on which a latest-version
1905 3000 7 ! scan is in progress, always read the entire directory.
1906 3001 7
1907 3002 7 D[D_BUF_LEN] = .D[D_DIR_LEN] * 512;
1908 3003 7 IF
1909 3004 7 .D[D_BUF_LEN] GTRU DIR_BUF_COUNT*512 AND
1910 3005 8 NOT (.DIR_STRUCLEV EQL 1 AND .DIR_SEL_LATEST LEQ 0)
1911 3006 7 THEN
1912 3007 7 D[D_BUF_LEN] = DIR_BUF_COUNT*512;
1913 3008 7
```



```
: 1914      3009 7
: 1915      3010 7
: 1916      3011 7
: 1917      3012 7
: 1918      3013 6
: 1919      3014 5
: 1920      3015 5
: 1921      3016 5
: 1922      3017 5
: 1923      3018 5
: 1924      3019 5
: 1925      3020 5
: 1926      3021 5
: 1927      3022 6
: 1928      3023 6
: 1929      3024 6
: 1930      3025 6
: 1931      3026 6
: 1932      3027 6
: 1933      3028 6
: 1934      3029 6
: 1935      3030 6
: 1936      3031 6
: 1937      3032 6
: 1938      3033 6
: 1939      3034 6
: 1940      3035 6
: 1941      3036 6
: 1942      3037 6
: 1943      3038 6
: 1944      3039 6
: 1945      3040 6
: 1946      3041 6
: 1947      3042 6
: 1948      3043 6
: 1949      3044 7
: 1950      3045 7
: 1951      3046 7
: 1952      3047 7
: 1953      3048 7
: 1954      3049 7
: 1955      3050 7
: 1956      3051 7
: 1957      3052 7
: 1958      3053 7
: 1959      3054 7
: 1960      3055 7
: 1961      3056 7
: 1962      3057 7
: 1963      3058 7
: 1964      3059 7
: 1965      3060 7
: 1966      3061 7
: 1967      3062 7
: 1968      3063 7
: 1969      3064 7
: 1970      3065 7

! Allocate memory for buffer.
!
D[D_BUF_ADDR] = GET_VM(D[D_BUF_LEN]);
END;

END;

! If we are not yet beyond the end, size the portion of the
! directory to be read on this pass and read it.
!
IF D[D_VBN] LEQU D[D_DIR_LEN]
THEN
  BEGIN
    LOCAL
      READ_ADDRESS,      ! Address for current read QIO
      PROC_LENGTH;       ! Bytes to process this time

    ! Compute size of chunk to be processed on this iteration.
    !
    PROC_LENGTH = MINU(
      D[D_BUF_LEN],
      (D[D_DIR_LEN] - D[D_VBN] + 1) * 512);
    D[D_BUF_LIM] = D[D_BUF_ADDR] + .PROC_LENGTH;
    D[D_REC] = D[D_VER] = 0;

    ! Loop to read the chunk of directory that will be processed on
    ! this iteration. Because the chunk might exceed 65535 bytes,
    ! we must have a loop, though it is unlikely to be executed
    ! more than once.
    !
    READ_ADDRESS = D[D_BUF_ADDR];
    WHILE .READ_ADDRESS -LSSA D[D_BUF_LIM] DO
      BEGIN
        LOCAL
          READ_LENGTH;    ! Size of current transfer in bytes

        ! Compute size of this transfer.
        !
        READ_LENGTH = MINU(
          127 * 512,
          D[D_BUF_LIM] - .READ_ADDRESS);

        ! Read in the blocks.
        !
        STATUS = C$QIOW(
          FUNC=IOS_READVBLK,
          CHAN=.DIR_CHAN,
          IOSB=IOSB,
          P1=.READ_ADDRESS,
          P2=.READ_LENGTH,
          P3=D[D_VBN]);
        READ_ADDRESS = .READ_ADDRESS + .READ_LENGTH;
```

```

: 1971      3066 7      D[D_VBN] = .D[D_VBN] + .READ_LENGTH/512;
: 1972      3067 7      IF .STATUS THEN STATUS = .IOSB[0];
: 1973      3068 7      IF NOT .STATUS
: 1974      3069 7      THEN
: 1975      3070 8          BEGIN
: 1976      3071 8              SIGNAL(
: 1977      3072 8                  BACKUP$_READDIR,
: 1978      3073 8                      2,
: 1979      3074 8                      .DIR_DEV_DESC,
: 1980      3075 8                      (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
: 1981      3076 8                      .STATUS);
: 1982      3077 8
: 1983      3078 8
: 1984      3079 8          ! Readjust context so that processing of the directory
: 1985      3080 8          ! is aborted.
: 1986      3081 8
: 1987      3082 8          D[D_DIR_LEN] = 0;
: 1988      3083 8          D[D_BUF_LIM] = 0;
: 1989      3084 8          EXITLOOP;
: 1990      3085 7          END;
: 1991      3086 6      END;
: 1992      3087 5      END;
: 1993      3088 5
: 1994      3089 5
: 1995      3090 5      ! Deaccess the directory file.
: 1996      3091 5      !
: 1997      3092 5      P C$QIOW(
: 1998      3093 5      P     FUNC=IOS$ DEACCESS,
: 1999      3094 5          CHAN=.DIR_CHAN);
: 2000      3095 4      END;
: 2001      3096 3      END;
: 2002      3097 3
: 2003      3098 3
: 2004      3099 3      ! If no more directory is available, pop up to the higher directory.
: 2005      3100 3      !
: 2006      3101 3      IF .D[D_REC] GEQA .D[D_BUF_LIM]
: 2007      3102 3      THEN
: 2008      3103 4          BEGIN
: 2009      3104 4
: 2010      3105 4          ! Deallocate dynamic memory.
: 2011      3106 4          !
: 2012      3107 4          IF .D[D_BUF_ADDR] NEQ 0
: 2013      3108 4          THEN
: 2014      3109 4              FREE_VM(.D[D_BUF_LEN], .D[D_BUF_ADDR]);
: 2015      3110 4          IF .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER] NEQ 0
: 2016      3111 4          THEN
: 2017      3112 4              FREE_VM(DIR$$NAME, .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER]);
: 2018      3113 4          IF .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER] NEQ 0
: 2019      3114 4          THEN
: 2020      3115 4              FREE_VM(DIR$$NAME, .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
: 2021      3116 4
: 2022      3117 4
: 2023      3118 4          ! Pop the directory stack.
: 2024      3119 4          !
: 2025      3120 4          DIR_STRING[0] = .D[D_SAV_LEN];
: 2026      3121 4          CH$FILL(0, D_S_ENTRY, .DT);
: 2027      3122 4          DIR_SP = D = .D - D_S_ENTRY;
```



```

2028 3123 4 DIR_LEVELS = .DIR_LEVELS - 1;
2029 3124 4
2030 3125 4
2031 3126 4 ! If the directory stack is now empty, we have completed the MFD.
2032 3127 4 !
2033 3128 4 IF .DIR_LEVELS LEQ 0
2034 3129 4 THEN
2035 3130 5 BEGIN
2036 3131 5 N[NAM$B_RSL] = 0;
2037 3132 5 N[NAM$W_FID_NUM] = N[NAM$W_FID_SEQ] = N[NAM$W_FID_RVN] = 0;
2038 3133 5 N[NAM$W_DID_NUM] = N[NAM$W_DID_SEQ] = N[NAM$W_DID_RVN] = 0;
2039 3134 5 RETURN FALSE;
2040 3135 4 END;
2041 3136 4
2042 3137 4 END;
2043 3138 4
2044 3139 4 ! Adjust pointers to the next entry in the directory.
2045 3140 4 !
2046 3141 4 R = .D[D_REC];
2047 3142 4 V = .D[D_VER];
2048 3143 4 IF TESTBITCC(DIR_FLAGS[D_SCAN_FAILED])
2049 3144 4 THEN
2050 3145 5 BEGIN
2051 3146 5 IF .DIR_STRUCLEV EQL 1
2052 3147 5 THEN
2053 3148 6 BEGIN
2054 3149 6 WHILE TRUE DO
2055 3150 6 BEGIN
2056 3151 6
2057 3152 6 ! Advance the record pointer to the next (or first) entry.
2058 3153 6 !
2059 3154 6 IF .R EQL 0
2060 3155 6 THEN
2061 3156 6 R = .D[D_BUF_ADDR]
2062 3157 6 ELSE
2063 3158 7 BEGIN
2064 3159 7 R = .R + NMB$C_DIRENTRY;
2065 3160 7 V = .V + 1;
2066 3161 6 END;
2067 3162 6
2068 3163 6
2069 3164 6 ! If there are no more entries, exit the loop.
2070 3165 6 !
2071 3166 6 IF .R GEQA .D[D_BUF_LIM] THEN EXITLOOP;
2072 3167 6
2073 3168 6
2074 3169 6 ! If the entry contains a non-zero file number, it is in use.
2075 3170 6 !
2076 3171 6 IF .R[NMB$W_FID_NUM] NEQ 0
2077 3172 6 THEN
2078 3173 7 BEGIN
2079 3174 7
2080 3175 7 ! If the selection filespec selects latest version,
2081 3176 7 ! determine if this is the latest version, by scanning
2082 3177 7 ! the entire directory for a higher version.
2083 3178 7 !
2084 3179 7 D[D_VER_COUNT] = 0;

```

```
2085 3180 7 IF .DIR_SEL_LATEST LEQ 0
2086 3181 7 THEN
2087 3182 7   INCRA P
2088 3183 7   FROM .D[D_BUF_ADDR]
2089 3184 7   TO .D[D_BUF_ADDR] + .D[D_BUF_LEN] - NMBSC_DIRENTRY
2090 3185 7   BY NMBSC_DIRENTRY DO
2091 3186 8     BEGIN
2092 3187 8     MAP
2093 3188 8       P: REF BBLOCK;
2094 3189 8
2095 3190 8     IF
2096 3191 8       .(P[NMB$W_NAME]) EQL .(R[NMB$W_NAME]) AND
2097 3192 8       .(P[NMB$W_NAME]+4) EQL .(R[NMB$W_NAME]+4) AND
2098 3193 8       .P[NMB$W_VERSION] GTRU .R[NMB$W_VERSION]
2099 3194 8     THEN
2100 3195 9       BEGIN
2101 3196 9       D[D_VER_COUNT] = -32768;
2102 3197 9       EXITLOOP;
2103 3198 8       END;
2104 3199 7     END;
2105 3200 7
2106 3201 7   ! Exit the loop with a valid entry, and D_VER_COUNT set.
2107 3202 7   !
2108 3203 7   EXITLOOP;
2109 3204 7   END;
2110 3205 6 END;
2111 3206 5
2112 3207 5 END
2113 3208 4 ELSE
2114 3209 5 BEGIN
2115 3210 5   WHILE TRUE DO
2116 3211 6     BEGIN
2117 3212 6     LOCAL
2118 3213 6     NEXT_RECORD;
2119 3214 6     ! Pointer to next record
2120 3215 6
2121 3216 6     ! Advance the record and version pointers to the
2122 3217 6     ! next (or first) entry.
2123 3218 6     !
2124 3219 6     IF .R EQL 0
2125 3220 6     THEN
2126 3221 6       R = .D[D_BUF_ADDR]
2127 3222 6     ELSE
2128 3223 6       IF .V NEQ 0
2129 3224 6       THEN
2130 3225 7         BEGIN
2131 3226 7         NEXT_RECORD = .R[DIR$W_SIZE] + .R + 2;
2132 3227 7         V = .V + DIR$C_VERSION;
2133 3228 7         IF .V LEQA .NEXT_RECORD - DIR$C_VERSION THEN EXITLOOP;
2134 3229 7         R = .NEXT_RECORD;
2135 3230 6         END;
2136 3231 6
2137 3232 6     ! If there are no more entries, exit the loop.
2138 3233 6     !
2139 3234 6     IF .R GEQA .D[D_BUF_LIM] THEN EXITLOOP;
2140 3235 6
2141 3236 6
```



```
2142 3237 6
2143 3238 6
2144 3239 6
2145 3240 7
2146 3241 7
2147 3242 7
2148 3243 7
2149 3244 7
2150 3245 7
2151 3246 7
2152 3247 6
2153 3248 7
2154 3249 7
2155 3250 7
2156 3251 7
2157 3252 7
2158 3253 7
2159 3254 7
2160 3255 8
2161 3256 8
2162 3257 8
2163 3258 8
2164 3259 8
2165 3260 8
2166 3261 8
2167 3262 8
2168 3263 8
2169 3264 8
2170 3265 9
2171 3266 9
2172 3267 9
2173 3268 9
2174 3269 9
2175 3270 9
2176 3271 8
2177 3272 7
2178 3273 8
2179 3274 8
2180 3275 8
2181 3276 8
2182 3277 8
2183 3278 8
2184 3279 8
2185 3280 8
2186 3281 8
2187 3282 8
2188 3283 8
2189 3284 8
2190 3285 8
2191 3286 8
2192 3287 8
2193 3288 7
2194 3289 7
2195 3290 7
2196 3291 7
2197 3292 7
2198 3293 7

IF .R[DIR$W_SIZE] EQL 65535
THEN
  BEGIN
    ! End of this block. Advance to next.
    R = ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512;
    V = 0;
  END
ELSE
  BEGIN
    ! Point to where next record should start. Make some
    ! validity tests on the entry we are looking at.
    NEXT_RECORD = .R[DIR$W_SIZE] + .R + 2;
    IF
      BEGIN
        IF
          .NEXT_RECORD GEQA ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512 OR
          ! Entry within block?
          .R[DIR$W_SIZE] OR
          ! Length even?
          .R[DIR$W_SIZE] LSSU DIR$C_LENGTH + DIR$C_VERSION
          ! Long enough?
        THEN
          TRUE
        ELSE
          BEGIN
            V = (.R + DIR$C_LENGTH + .R[DIR$B_NAMECOUNT] + 1) AND NOT 1;
            ! Proper type code?
            .R[DIR$V_TYPE] NEQ DIR$C_FID OR
            ! Version entry within block?
            .V GEQA ((.R - .D[D_BUF_ADDR]) AND NOT 511) + .D[D_BUF_ADDR] + 512 - DIR$C_VERSION
          END
        END
      END
    THEN
      BEGIN
        ! Directory format error. Report it and quit.
        !
        SIGNAL(
          BACKUP$_BADDR,
          2,
          DIR_DEV_DESC,
          (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING));

        ! Adjust context to avoid processing this directory.
        R = .D[D_BUF_LIM];
        D[D_DIR_LEN] = 0;
      END
    ! Found a valid entry. Exit the loop.
    EXITLOOP;
```

```
2199 3294 6      END;
2200 3295 5      END;
2201 3296 4      END;
2202 3297 4      D[D_REC] = .R;
2203 3298 4      D[D_VER] = .V;
2204 3299 3      END;
2205 3300 3
2206 3301 3
2207 3302 3      ! If an entry was found, finish processing it.
2208 3303 3
2209 3304 3      IF .R LSSA .D[D_BUF_LIM]
2210 3305 3      THEN
2211 3306 4          BEGIN
2212 3307 4              LOCAL
2213 3308 4                  RSA_DESC: VECTOR[2],      ! Descriptor for RSA area
2214 3309 4                  NAME: REF VECTOR[.BYTE],    ! Pointer to ASCII name.typ
2215 3310 4                  VERSION,                  ! Binary version number
2216 3311 4                  FILE_NAME: VECTOR[20,BYTE]; ! Area to build ODS-1 filename
2217 3312 4
2218 3313 4
2219 3314 4      ! Get a pointer to the name.typ string and the version number.
2220 3315 4      ! Initialize the file ID in the name block.
2221 3316 4
2222 3317 4      IF .DIR_STRUCLEV EQL 2
2223 3318 4      THEN
2224 3319 5          BEGIN
2225 3320 5              D[D_VER_COUNT] = .D[D_VER_COUNT] - 1;
2226 3321 5              IF
2227 3322 5                  .R[DIR$B_NAMECOUNT] NEQ .BBLOCK[D[D_NAME_DESC], DSC$W_LENGTH] OR
2228 3323 5                  CH$NEQ(
2229 3324 5                      .R[DIR$B_NAMECOUNT],
2230 3325 5                      R[DIR$T_NAME],
2231 3326 5                      .R[DIR$B_NAMECOUNT],
2232 3327 5                      .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER])
2233 3328 5              THEN
2234 3329 6                  BEGIN
2235 3330 6                      D[D_VER_COUNT] = 0;
2236 3331 6                      BBLOCK[D[D_NAME_DESC], DSC$W_LENGTH] = .R[DIR$B_NAMECOUNT];
2237 3332 6                      CH$MOVE(
2238 3333 6                          .R[DIR$B_NAMECOUNT],
2239 3334 6                          R[DIR$T_NAME],
2240 3335 6                          .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
2241 3336 5                  END;
2242 3337 5                  NAME = R[DIR$B_NAMECOUNT];
2243 3338 5                  VERSION = .V[DIR$W_VERSION];
2244 3339 5                  DIR_VERLIMIT = .R[DIR$W_VERLIMIT];
2245 3340 5                  N[NAM$W_FID_NUM] = .V[DIR$W_FID_NUM];
2246 3341 5                  N[NAM$W_FID_SEQ] = .V[DIR$W_FID_SEQ];
2247 3342 5                  N[NAM$W_FID_RVN] = .V[DIR$W_FID_RVN];
2248 3343 5                  IF .N[NAM$B_FID_RVN] EQL 0
2249 3344 5                      THEN N[NAM$B_FID_RVN] = .BBLOCK[D[D_FID], FID$B_RVN];
2250 3345 5                  END
2251 3346 4      ELSE
2252 3347 5          BEGIN LOCAL T;
2253 3348 5              T = MAKE_STRING(.R, FILE_NAME[1]);
2254 3349 5              FILE_NAME[0] = CH$FIND_CR(T, FILE_NAME[1], %C';') - FILE_NAME[1];
2255 3350 5              NAME = FILE_NAME;
```



```
2256 3351 5      VERSION = .R[NMBSW_VERSION];
2257 3352 5      N[NAMSW_FID_NUM] = .R[NMBSW_FID_NUM];
2258 3353 5      N[NAMSW_FID_SEQ] = .R[NMBSW_FID_SEQ];
2259 3354 5      N[NAMSW_FID_RVN] = 1;
2260 3355 4      END;
2261 3356 4
2262 3357 4
2263 3358 4      ! Initialize the resultant string in the name block.
2264 3359 4      !
2265 3360 4      RSA_DESC[0] = .N[NAMSB_RSS];
2266 3361 4      RSA_DESC[1] = .N[NAMSL_RSA];
2267 3362 4      $FAO(
P      $DESCRIPTOR('!AS[!AC]!AC;!UW'),
P      RSA_DESC,
P      RSA_DESC,
P      .DIR_DEV_DESC,
P      (IF .DIR_STRING[0] EQL 0 THEN MFD ELSE DIR_STRING),
P      .NAME,
2274 3369 4      .VERSION);
2275 3370 4      N[NAMSB_RSL] = .RSA_DESC[0];
2276 3371 4      INIT_NAMEBLOCK(.N);
2277 3372 4
2278 3373 4      ! Initialize the directory ID in the name block.
2279 3374 4      !
2280 3375 4      N[NAMSW_DID_NUM] = .BBLOCK[D[D_FID], FIDSW_NUM];
2281 3376 4      N[NAMSW_DID_SEQ] = .BBLOCK[D[D_FID], FIDSW_SEQ];
2282 3377 4      N[NAMSW_DID_RVN] = .BBLOCK[D[D_FID], FIDSW_RVN];
2283 3378 4
2284 3379 4
2285 3380 4
2286 3381 4      ! Set the directory scan bits for the next iteration.
2287 3382 4      !
2288 3383 4      DIR_STATUS = 0;
2289 3384 4      IF
2290 3385 4      .DIR_LEVELS LEQ D_K_NLEVELS-1 AND
2291 3386 5      (IF .N[NAMSB_TYPE] EQL 4
2292 3387 5      THEN .N[NAMSL_TYPE] EQL '.DIR'
2293 3388 4      ELSE FALSE) AND
2294 3389 4      .VERSION EQL 1 AND
2295 3390 5      (.N[NAMSW_FID_NUM] NEQ FIDSC_MFD OR
2296 3391 5      .N[NAMSB_FID_NMX] NEQ 0)
2297 3392 4      THEN
2298 3393 5      BEGIN
2299 3394 5      LOCAL
2300 3395 5      STATUS,      ! Status variable
2301 3396 5      DIR_DESC: VECTOR[2]; ! Descriptor for new directory string
2302 3397 5
2303 3398 5
2304 3399 5      ! Note that this is a directory.
2305 3400 5      !
2306 3401 5      DIR_STATUS[D_STAT_VALID] = TRUE;
2307 3402 5
2308 3403 5
2309 3404 5      ! Generate the new directory string. If this is not the top level,
2310 3405 5      ! append a dot, and then append the directory name.
2311 3406 5      !
2312 3407 5      DIR_DESC[0] = .DIR_STRING[0];
```



```
2313 3408 S DIR_DESC[1] = DIR_STRING[1];
2314 3409 S IF .DIR_DESC[0] NEQ 0
2315 3410 S THEN
2316 3411 S BEGIN
2317 3412 S DIR_DESC[0] = .DIR_DESC[0] + 1;
2318 3413 S DIR_STRING[.DIR_DESC[0]] = %C'.';
2319 3414 S END;
2320 3415 S CH$MOVE(
2321 3416 S .N[NAM$B_NAME],
2322 3417 S .N[NAM$L_NAME],
2323 3418 S DIR_STRING[.DIR_DESC[0]+1]);
2324 3419 S DIR_DESC[0] = .DIR_DESC[0] + .N[NAM$B_NAME];
2325 3420 S
2326 3421 S ! Temporarily push down the directory stack.
2327 3422 S !
2328 3423 S D = .D + D_S_ENTRY;
2329 3424 S
2330 3425 S
2331 3426 S ! Allocate the dynamic areas for this level.
2332 3427 S !
2333 3428 S BBLOCK[D[D_TERM_DESC], DSC$A_POINTER] = GET_VM(DIR$S_NAME);
2334 3429 S BBLOCK[D[D_NAME_DESC], DSC$A_POINTER] = GET_VM(DIR$S_NAME);
2335 3430 S
2336 3431 S
2337 3432 S ! Match the directory specification against the pattern string to
2338 3433 S ! determine if this directory should be scanned. If it should not,
2339 3434 S ! pop the directory stack. Otherwise, finish initializing context.
2340 3435 S !
2341 3436 S IF .DIR_FLAGS[D_IMAGE_SCAN]
2342 3437 S THEN
2343 3438 S STATUS = %B'111'
2344 3439 S ELSE
2345 3440 S STATUS = MATCH_DIRECTORY(
2346 3441 S DIR_DESC,
2347 3442 S DIR_SEL_DIR,
2348 3443 S D[D_TERM_DESC],
2349 3444 S D[D_TERM_VER],
2350 3445 S DIR_SEL_NTV);
2351 3446 S
2352 3447 S
2353 3448 S IF .STATUS<1,1>
2354 3449 S THEN
2355 3450 S BEGIN
2356 3451 S DIR_SP[D DIR_SCAN] = DIR_STATUS[D STAT_SCANNED] = TRUE;
2357 3452 S IF .STATUS<0,1> THEN D[D_DIR_MATCHES] = DIR_STATUS[D_STAT_FILE_SEL] = TRUE;
2358 3453 S IF .STATUS<2,1> THEN D[D_WILD_TERM] = TRUE;
2359 3454 S IF .STATUS<3,1> THEN D[D_NON_TERM] = TRUE;
2360 3455 S END
2361 3456 S ELSE
2362 3457 S BEGIN
2363 3458 S FREE_VM(DIR$S_NAME, .BBLOCK[D[D_TERM_DESC], DSC$A_POINTER]);
2364 3459 S FREE_VM(DIR$S_NAME, .BBLOCK[D[D_NAME_DESC], DSC$A_POINTER]);
2365 3460 S CH$FILL(0, D_S_ENTRY, .D);
2366 3461 S END;
2367 3462 S
2368 3463 S
2369 3464 S
```



```
2370      3465 5      ! Pop the directory stack.
2371      3466 5      !
2372      3467 5      D = .DIR_SP;
2373      3468 4      END;
2374      3469 4
2375      3470 4
2376      3471 4      ! If this is image mode, the file always matches.
2377      3472 4      !
2378      3473 4      IF
2379      3474 4          .DIR_FLAGS[D_IMAGE_SCAN] AND
2380      3475 5          (NOT .DIR_FLAGS[D_SCANNED_DIRS] OR
2381      3476 5              .N[NAM$W_FID_NUM] NEQ FID$C_MFD OR
2382      3477 5              .N[NAM$B_FID_NMX] NEQ 0)
2383      3478 4      THEN
2384      3479 5          BEGIN
2385      3480 5              IF .DIR_STATUS[D_STAT_VALID]
2386      3481 5                  THEN DIR_STATUS[D_STAT_DIR_SEL] = TRUE;
2387      3482 5          EXITLOOP;
2388      3483 4          END;
2389      3484 4
2390      3485 4
2391      3486 4      ! Test for scan termination.
2392      3487 4      !
2393      3488 4      IF
2394      3489 5          BEGIN
2395      3490 5              IF .DIR_STRUCLEV EQL 1
2396      3491 5                  THEN
2397      3492 5                      .V GTRU .DIR_SCANLIMIT[.DIR_LEVELS-1]
2398      3493 5                  ELSE
2399      3494 5                      IF .BBLOCK[D_TERM_DESC], DSC$W_LENGTH EQL 0
2400      3495 5                          THEN
2401      3496 5                              FALSE
2402      3497 5                          ELSE
2403      3498 5                              TERMINATE_SCAN(
2404      3499 5                                  .NAME[0], NAME[1],
2405      3500 5                                  .VERSION,
2406      3501 5                                  .BBLOCK[D_TERM_DESC], DSC$W_LENGTH,
2407      3502 5                                  .BBLOCK[D_TERM_DESC], DSC$A_POINTER,
2408      3503 5                                  .D[D_TERM_VER])
2409      3504 5                              END
2410      3505 4                      THEN
2411      3506 5                          BEGIN
2412      3507 5                              IF .DIR_FLAGS[D_HARD_STOP]
2413      3508 5                                  THEN
2414      3509 6                                  BEGIN
2415      3510 6                                      DIR_FLAGS[D_SCAN_FAILED] = TRUE;
2416      3511 6                                      RETURN FALSE;
2417      3512 5                                  END;
2418      3513 5
2419      3514 5
2420      3515 5      ! Adjust context to skip rest of this directory.
2421      3516 5      !
2422      3517 5      D[D_REC] = .D[D_BUF_LIM];
2423      3518 5      D[D_DIR_LEN] = 0;
2424      3519 4      END;
2425      3520 4
2426      3521 4
```

```
: 2427      3522  4      ! Match the selection file specification with the resultant string.
: 2428      3523  4      !
: 2429      3524  4      IF
: 2430      3525  4      .D[D_REC] LSSA .D[D_BUF_LIM] AND
: 2431      3526  4      .D[D_DIR_MATCHES] AND
: 2432      3527  4      (.DIR_SEL_LATEST GTR 0 OR .DIR_SEL_LATEST EQL .D[D_VER_COUNT]) AND
: 2433      3528  5      (NOT .DIR_FLAGS[D_SCANNED_DIRS] OR
: 2434      3529  5      .N[NAM$W_FID_NUM] NEQ FID$C_MFD OR
: 2435      3530  5      .N[NAM$B_FID_NMX] NEQ 0)
: 2436      3531  4      THEN
: 2437      3532  5      BEGIN
: 2438      3533  5      LOCAL
: 2439      3534  5      NTV_DESC:      VECTOR[2];      ! Descriptor for n.t;v
: 2440      3535  5
: 2441      3536  5
: 2442      3537  5      ! Match the file specification.
: 2443      3538  5      !
: 2444      3539  5      NTV_DESC[0] = .N[NAM$L_RSA] + .N[NAM$B_RSL] - .N[NAM$L_NAME];
: 2445      3540  5      NTV_DESC[1] = .N[NAM$L_NAME];
: 2446      3541  5      IF MATCH_FILENAME(NTV_DESC, DIR_SEL_NTV)
: 2447      3542  5      THEN
: 2448      3543  6      BEGIN
: 2449      3544  6      IF .DIR_STATUS[D_STAT_VALID]
: 2450      3545  6      THEN DIR_STATUS[D_STAT_DIR_SEL] = TRUE;
: 2451      3546  6      EXITLOOP;
: 2452      3547  5      END;
: 2453      3548  4      END;
: 2454      3549  4
: 2455      3550  4
: 2456      3551  4      ! If scanned directories are requested, and the directory was not
: 2457      3552  4      ! selected above, return it anyway.
: 2458      3553  4      !
: 2459      3554  4      IF .DIR_FLAGS[D_SCANNED_DIRS] AND .D[D_DIR_SCAN] AND .D[D_NON_TERM]
: 2460      3555  4      THEN
: 2461      3556  4      EXITLOOP;
: 2462      3557  3      END;
: 2463      3558  2      END;
: 2464      3559  2
: 2465      3560  2
: 2466      3561  2      TRUE
: 2467      3562  1      END;
```

```
57 55 21 3B 53 41 21 00F5D P.AAJ: .ASCII \!AS;!UW\
00000007 00F64 P.AAI: .LONG 7
00000000 00F68 .ADDRESS P.AAJ
57 55 21 3B 43 41 21 5D 43 41 21 5B 53 41 21 00F6C P.AAL: .ASCII \!AS[!AC]!AC;!UW\
00F7B .BLKB 1
0000000F 00F7C P.AAK: .LONG 15
00000000 00F80 .ADDRESS P.AAL
.EXTRN STA_QIOW
OFFC 00000 .ENTRY FIND_NEXT, Save R2,R3,R4,R5,R6,R7,R8,R9,-
SE FEB4 CE 9E 00002 MOVAB R10,R11 -332(SP), SP
: 2668
```



		58	00000000'	EF	D0	00007	MOVL	DIR_NAM, N	2703	
		56	00000000'	EF	D0	0000E	MOVL	DIR_SP, D	2704	
			00000000'	EF	B4	00015	CLRW	DIR-VERLIMIT	2705	
			00000000'	EF	94	0001B	CLRB	DIR-STATUS		
			00000000'	EF	95	00021	TSTB	DIR-LEVELS	2710	
				03	12	00027	BNEQ	1\$		
		0490		31	00029	BRW	44\$			
17	00000000'	EF		04	E5	0002C	1\$:	BBCC	#4, DIR_FLAGS, 2\$	2722
0F	00000000'	EF		03	E1	00034		BBC	#3, DIR_FLAGS, 2\$	2724
0A	23	A6		03	E1	0003C		BBC	#3, 35(D), 2\$	
	00000000'	EF		05	88	00041		BISB2	#5, DIR-STATUS	2727
				08B7	31	00048		BRW	96\$	2728
4D	00000000'	EF		01	E0	0004B	2\$:	BBS	#1, DIR_FLAGS, 4\$	2745
45	00000000'	EF		05	E0	00053		BBS	#5, DIR_FLAGS, 4\$	
40	23	A6		00	E5	0005B		BBCC	#0, 35(D), 4\$	2746
			00000000'	EF	96	00060		INCB	DIR-LEVELS	2752
		56	44	A6	9E	00066		MOVAB	68(R6), D	2753
	00000000'	EF		56	D0	0006A		MOVL	D, DIR_SP	
		50	1C	A6	9E	00071		MOVAB	28(D), -R0	2754
		60	24	A8	D0	00075		MOVL	36(N), (R0)	
	04	A0	28	A8	B0	00079		MOVW	40(N), 4(R0)	2756
	22	A6			EF	90	0007E	MOVB	DIR-STRING, 34(D)	2757
	14	A6			01	D0	00086	MOVL	#1, 20(D)	2758
			00000000'	EF	95	0008A		TSTB	DIR-STRING	2764
				06	13	00090		BEQL	3\$	
			00000000'	EF	96	00092		INCB	DIR-STRING	
	00000000'	EF	3B	A8	80	00098	3\$:	ADDB2	59(N), DIR-STRING	2765
		5A	10	A6	9E	000A0	4\$:	MOVAB	16(D), R10	2772
		6A		66	D1	000A4		CMPL	(D), (R10)	
				03	1E	000A7		BGEQU	6\$	
				03B3	31	000A9	5\$:	BRW	40\$	
		01	14	A6	D1	000AC	6\$:	CMPL	20(D), #1	2773
				07	13	000B0		BEQL	7\$	
	18	A6	14	A6	D1	000B2		CMPL	20(D), 24(D)	
				F0	1A	000B7		BGTRU	5\$	
	B8	AD	40	8F	9A	000B9	7\$:	MOVZBL	#64, FIB-DESC	2791
	BC	AD	C0	AD	9E	000BE		MOVAB	FIB, FIB-DESC+4	2792
0040	8F			00	00	2C	000C3	MOVCS	#0, (SP), #0, #64, FIB	2793
			C0	AD		000CA				
			00200000	8F	D0	000CC		MOVL	#2097152, FIB	2794
08	00000000'	EF		02	E1	000D4		BBC	#2, QUAL+10, 8\$	2795
		AD	00300000	8F	D0	000DC		MOVL	#3145728, FIB	
		57	1C	A6	9E	000E4	8\$:	MOVAB	28(D), R7	2796
				67	D0	000E8		MOVL	(R7), FIB+4	
	C4	AD		A7	B0	000EC		MOVW	4(R7), FIB+8	2798
	C8	AD	04	8F	D0	000F1		MOVL	#655426, ATR-DESC	2799
	FF60	CD	000A0042	8F	D0	000FA		MOVAB	HEADER, ATR-DESC+4	2801
	FF64	CD	FF6C	CD	9E	000FA		CLRL	ATR-DESC+8	2802
			FF68	CD	D4	00101		MOVL	DIR-CHAN, R0	2808
		50	00000000'	EF	D0	00105		CMPL	R0, -#131071	
	0001FFFF	8F		50	D1	0010C		BLSSU	9\$	
				23	1F	00113		CLRL	-(SP)	
			FF60	7E	D4	00115		PUSHAB	ATR-DESC	
				7E	7C	00117		CLRL	-(SP)	
				7E	D4	0011D		CLRL	-(SP)	
		B8		AD	9F	0011F		PUSHAB	FIB-DESC	
				7E	7C	00122		CLRL	-(SP)	

				B0	AD	9F	00124		PUSHAB	IOSB		
		7E		72	8F	9A	00127		MOVZBL	#114, -(SP)		
					50	DD	0012B		PUSHL	R0		
					7E	D4	0012D		CLRL	-(SP)		
		00000000G	00		0C	FB	0012F		CALLS	#12, STA_QIOW		
					21	11	00136		BRB	10\$		
					7E	D4	00138	9\$:	CLRL	-(SP)		
				FF60	CD	9F	0013A		PUSHAB	ATR_DESC		
					7E	7C	0013E		CLRQ	-(SP)		
					7E	D4	00140		CLRL	-(SP)		
				B8	AD	9F	00142		PUSHAB	FIB_DESC		
					7E	7C	00145		CLRQ	-(SP)		
				B0	AD	9F	00147		PUSHAB	IOSB		
			7E	72	8F	9A	0014A		MOVZBL	#114, -(SP)		
					50	DD	0014E		PUSHL	R0		
					7E	D4	00150		CLRL	-(SP)		
		00000000G	9F		0C	FB	00152		CALLS	#12, @#SYSSQIOW		
			59		50	DD	00159	10\$:	MOVL	R0, STATUS		2809
			04		59	E9	0015C		BLBC	STATUS, 11\$		
			59		B0	AD	3C	0015F	MOVZWL	IOSB, STATUS		2817
			24		59	B1	00163	11\$:	CMPW	STATUS, #36		
					08	12	00166		BNEQ	12\$		2818
		03 00000000'	EF		01	E1	00168		BBC	#1, COM_FLAGS, 13\$		
					00E3	31	00170	12\$:	BRW	16\$		2819
		F8	23	A6	02	E0	00173	13\$:	BBS	#2, 35(D), 12\$		2820
					0C	A6	D5	00178	TSTL	12(D)		
					F3	12	0017B		BNEQ	12\$		2830
			08	A6	0200	8F	3C	0017D	MOVZWL	#512, 8(D)		2831
			7E		0200	8F	3C	00183	MOVZWL	#512, -(SP)		
		00000000G	00		01	FB	00188		CALLS	#1, GET_VM		
			0C	A6	50	DD	0018F		MOVL	R0, 12(D)		
			00A4	CE	40	8F	9A	00193	MOVZBL	#64, FIB_DESC		2836
			FF5C	CD	64	AE	9E	00199	MOVAB	FIB, FIB_DESC+4		2837
0040	8F	00		6E	00	2C	0019F	MOVCS	#0, (SP), #0, #64, FIB		2838	
					64	AE	001A6					
			6E	AE	67	DD	001A8		MOVL	(R7), FIB+10		2839
			72	AE	04	A7	DD	001AC	MOVW	4(R7), FIB+14		2841
			5C	AE	56	8F	9A	001B1	MOVZBL	#86, FNA_DESC		2846
			60	AE	04	AE	9E	001B6	MOVAB	FNA, FNA_DESC+4		2847
				7E	38	A6	3C	001BB	MOVZWL	56(D), -(SP)		2852
				52	24	A6	9E	001BF	MOVAB	36(D), R2		
					52	DD	001C3		PUSHL	R2		
				64	AE	9F	001C5		PUSHAB	FNA_DESC		
				68	AE	9F	001C8		PUSHAB	FNA_DESC		
				FE11	CF	9F	001CB		PUSHAB	P.AXI		
		00000000G	00		05	FB	001CF		CALLS	#5, SYSSFAO		
					7E	7C	001D6		CLRQ	-(SP)		2862
					7E	7C	001D8		CLRQ	-(SP)		
				6C	AE	9F	001DA		PUSHAB	FNA_DESC		
				FF58	CD	9F	001DD		PUSHAB	FIB_DESC		
					7E	7C	001E1		CLRQ	-(SP)		
				B0	AD	9F	001E3		PUSHAB	IOSB		
					32	DD	001E6		PUSHL	#50		
				00000000'	EF	DD	001E8		PUSHL	DIR_CHAN		
					7E	D4	001EE		CLRL	-(SP)		
		00000000G	00		0C	FB	001F0		CALLS	#12, SYSSQIOW		
			59		50	DD	001F7		MOVL	R0, STATUS		



		04		59	E9	001FA	BLBC	STATUS, 14\$	2863
		59	B0	AD	3C	001FD	MOVZWL	IOSB, STATUS	2870
	0910	8F		59	B1	00201	CMPW	STATUS, #2320	2873
			18	08	12	00206	BNEQ	15\$	2874
		59		A6	D4	00208	CLRL	24(D)	2870
				01	D0	0020B	MOVL	#1, STATUS	2885
	00000000'	EF		46	11	0020E	BRB	16\$	2886
	14	A6		02	90	00210	MOVB	#2, DIR_STRUCLEV	2887
	18	A6		02	D0	00217	MOVL	#2, 20(D)	2892
		50		01	D0	0021B	MOVL	#1, 24(D)	2893
				62	3C	0021F	MOVZWL	(R2), R0	2894
		50		50	D6	00222	INCL	R0	2895
				01	8A	00224	BICB2	#1, L	2896
	63	53	0C	A6	D0	00227	MOVL	12(D), P	2897
		50		0C	A1	0022B	ADDW3	#12, L, (P)	2902
		02	A3	0090	CE	0022F	MOVW	FIB+44, 2(P)	2903
				04	A3	00235	CLRB	4(P)	2904
50		05	A3	62	90	00238	MOVB	(R2), 5(P)	2906
	00	04	B2	62	2C	0023C	MOVC5	(R2), #4(R2), #0, L, 6(P)	2907
				06	A3	00242			2902
		63	38	A6	B0	00244	MOVW	56(D), (P)	2903
		02	A3	68	AE	00248	MOVL	FIB+4, 2(P)	2904
		06	A3	6C	AE	0024D	MOVW	FIB+8, 6(P)	2906
			08	A3	9E	00252	MOVAB	8(R3), (R10)	2909
		35		59	E8	00256	BLBS	STATUS, 19\$	2914
				59	DD	00259	PUSHL	STATUS	2915
		00000000'		EF	95	0025B	TSTB	DIR_STRING	2924
				07	12	00261	BNEQ	17\$	2925
		50	EE15	CF	9E	00263	MOVAB	MFD, R0	2926
				07	11	00268	BRB	18\$	2927
		50	00000000'	EF	9E	0026A	MOVAB	DIR_STRING, R0	2928
				50	DD	00271	PUSHL	R0	2929
			00000000'	EF	DD	00273	PUSHL	DIR_DEV_DESC	2930
				02	DD	00279	PUSHL	#2	2931
			00000000G	8F	DD	0027B	PUSHL	#BACKUP\$ OPENDIR	2932
	00000000G	00		05	FB	00281	CALLS	#5, LIB\$SIGNAL	2933
			18	A6	D4	00288	CLRL	24(D)	2934
				01D1	31	0028B	BRW	40\$	2935
			0C	A6	D5	0028E	TSTL	12(D)	2936
				03	13	00291	BEQL	20\$	2937
				00A3	31	00293	BRW	27\$	2938
		50	18	A6	9E	00296	MOVAB	24(D), R0	2939
				60	D4	0029A	CLRL	(R0)	2940
	00000000'	EF	FF73	CD	90	0029C	MOVB	HEADER+7, DIR_STRUCLEV	2941
		02	00000000'	EF	91	002A5	CMPB	DIR_STRUCLEV, #2	2942
				22	12	002AC	BNEQ	22\$	2943
4E		A1	AD	05	E1	002AE	BBC	#5, HEADER+53, 24\$	2944
60		88	AD	10	9C	002B3	ROTL	#16, RECATR+8, (R0)	2945
			8C	AD	B5	002B8	TSTW	RECATR+12	2946
				02	12	002BB	BNEQ	21\$	2947
				60	D7	002BD	DECL	(R0)	2948
		3A	A6	AD	B0	002BF	MOVW	HEADER+64, 58(D)	2949
		3C	A6	AD	D0	002C4	MOVL	HEADER+60, 60(D)	2950
		40	A6	AD	B0	002C9	MOVW	HEADER+50, 64(D)	2951
				31	11	002CE	BRB	24\$	2952
		01	FF7A	CD	91	002D0	CMPB	RECATR, #1	2953
				2A	12	002D5	BNEQ	24\$	2954

		10	FF7C	CD	B1	002D7	CMPW	RECATTR+2, #16	2976
				23	12	002DC	BNEQ	24\$	2979
60	82	AD		10	9C	002DE	ROTL	#16, RECATTR+8, (R0)	2980
			86	AD	B5	002E3	TSTW	RECATTR+12	2981
				02	12	002E6	BNEQ	23\$	2982
				60	D7	002E8	DECL	(R0)	2983
	3A	A6	FF76	CD	B0	002EA	MOVW	HEADER+10, 58(D)	2984
	3C	A6	FF74	CD	9A	002F0	MOVZBL	HEADER+8, 60(D)	2985
	3E	A6	FF75	CD	9B	002F6	MOVZBW	HEADER+9, 62(D)	2993
	40	A6	98	AD	B0	002FC	MOVW	HEADER+44, 64(D)	3002
				60	D5	00301	TSTL	(R0)	3004
				34	13	00303	BEQL	27\$	3005
08	A6	00002000		09	78	00305	ASHL	#9, (R0), 8(D)	3007
			08	A6	D1	0030A	CMPL	8(D), #8192	3012
				17	1B	00312	BLEQU	26\$	3020
		01	00000000'	EF	91	00314	CMPB	DIR_STRUCLEV, #1	3032
			00000000'	08	12	0031B	BNEQ	25\$	3033
				06	D5	0031D	TSTL	DIR_SEL_LATEST	3034
				0F	15	00323	BLEQ	26\$	3042
	08	A6	2000	8F	3C	00325	MOVZWL	#8192, 8(D)	3043
			08	A6	DD	0032B	PUSHL	8(D)	3052
	00000000G	00		01	FB	0032E	CALLS	#1, GET_VM	3053
	0C	A6		50	D0	00335	MOVL	R0, 12(D)	3054
	18	A6	14	A6	D1	00339	CMPL	20(D), 24(D)	3055
				2B	1A	0033E	BGTRU	30\$	3056
50	18	A6	14	A6	C3	00340	SUBL3	20(D), 24(D), R0	3057
50		50		09	78	00346	ASHL	#9, R0, R0	3058
		51	0200	C0	9E	0034A	MOVAB	512(R0), R1	3059
		50	08	A6	D0	0034F	MOVL	8(D), R0	3060
		51		50	D1	00353	CMPL	R0, R1	3061
				03	1B	00356	BLEQU	28\$	3062
		50		51	D0	00358	MOVL	R1, R0	3063
	6A	0C	B640	9E	0035B	28\$:	MOVAB	012(D)[PROC_LENGTH], (R10)	3064
			66	7C	00360		CLRQ	(D)	3065
	53	0C	A6	D0	00362		MOVL	12(D), READ_ADDRESS	3066
	6A		53	D1	00366	29\$:	CMPL	READ_ADDRESS, (R10)	3067
			03	1F	00369		BLSSU	31\$	3068
			00B3	31	0036B	30\$:	BRW	38\$	3069
50		6A		53	C3	0036E	SUBL3	READ_ADDRESS, (R10), R0	3070
	0000FE00	8F		50	D1	00372	CMPL	R0, #65024	3071
				05	1B	00379	BLEQU	32\$	3072
		50	FE00	8F	3C	0037B	MOVZWL	#65024, R0	3073
		52		50	D0	00380	MOVL	R0, READ_LENGTH	3074
		50	00000000'	EF	D0	00383	MOVL	DIR_CHAN, R0	3075
	0001FFFF	8F		50	D1	0038A	CMPL	R0, #131071	3076
				1F	1F	00391	BLSSU	33\$	3077
				7E	7C	00393	CLRQ	-(SP)	3078
				7E	D4	00395	CLRL	-(SP)	3079
			14	A6	DD	00397	PUSHL	20(D)	3080
				52	DD	0039A	PUSHL	READ_LENGTH	3081
				53	DD	0039C	PUSHL	READ_ADDRESS	3082
				7E	7C	0039E	CLRQ	-(SP)	3083
			B0	AD	9F	003A0	PUSHAB	10SB	3084
				31	DD	003A3	PUSHL	#49	3085
				50	DD	003A5	PUSHL	R0	3086
				7E	D4	003A7	CLRL	-(SP)	3087
	00000000G	00		0C	FB	003A9	CALLS	#12, STA_Q10W	3088



		1D	11	003B0	BRB	34\$		
		7E	7C	003B2	CLRQ	-(SP)		
		7E	D4	003B4	CLRL	-(SP)		
	14	A6	DD	003B6	PUSHL	20(D)		
		52	DD	003B9	PUSHL	READ_LENGTH		
		53	DD	003BB	PUSHL	READ_ADDRESS		
		7E	7C	003BD	CLRQ	-(SP)		
	B0	AD	9F	003BF	PUSHAB	IOSB		
		31	DD	003C2	PUSHL	#49		
		50	DD	003C4	PUSHL	R0		
		7E	D4	003C6	CLRL	-(SP)		
00000000G	9F	0C	FB	003C8	CALLS	#12, @#SYSSQIOW		
	59	50	D0	003CF	MOVL	R0, STATUS		3065
	53	52	C0	003D2	ADDL2	READ_LENGTH, READ_ADDRESS		3066
	14	52	8F	C6	DIVL2	#512, R2		
	A6	52	C0	003DC	ADDL2	R2, 20(D)		
	0A	59	E9	003E0	BLBC	STATUS, 35\$		3067
	59	B0	AD	3C	MOVZWL	IOSB, STATUS		
	03	59	E9	003E7	BLBC	STATUS, 35\$		3068
		FF	79	31	BRW	29\$		
		59	DD	003ED	PUSHL	STATUS		3076
		00000000'	EF	95	TSTB	DIR_STRING		3075
		07	12	003F5	BNEQ	36\$		
	50	EC81	CF	9E	MOVAB	MFD, R0		
		07	11	003FC	BRB	37\$		
	50	00000000'	EF	9E	MOVAB	DIR_STRING, R0		
		50	DD	00405	PUSHL	R0		
		00000000'	EF	DD	PUSHL	DIR_DEV_DESC		3074
		02	DD	0040D	PUSHL	#2		3071
		00000000G	8F	DD	PUSHL	#BACKUP\$ READDIR		
00000000G	00	05	FB	00415	CALLS	#5, LIB\$SIGNAL		
		18	A6	D4	CLRL	24(D)		3082
		6A	D4	0041F	CLRL	(R10)		3083
	50	00000000'	EF	D0	MOVL	DIR_CHAN, R0		3094
0001FFFF	8F	50	D1	00428	CMPL	R0, #131071		
		18	1F	0042F	BLSSU	39\$		
		7E	7C	00431	CLRQ	-(SP)		
		7E	7C	00433	CLRQ	-(SP)		
		7E	7C	00435	CLRQ	-(SP)		
		7E	7C	00437	CLRQ	-(SP)		
	7E	34	7D	00439	MOVQ	#52, -(SP)		
		50	DD	0043C	PUSHL	R0		
		7E	D4	0043E	CLRL	-(SP)		
00000000G	00	0C	FB	00440	CALLS	#12, STA_QIOW		
		16	11	00447	BRB	40\$		
		7E	7C	00449	CLRQ	-(SP)		
		7E	7C	0044B	CLRQ	-(SP)		
		7E	7C	0044D	CLRQ	-(SP)		
		7E	7C	0044F	CLRQ	-(SP)		
	7E	34	7D	00451	MOVQ	#52, -(SP)		
		50	DD	00454	PUSHL	R0		
		7E	D4	00456	CLRL	-(SP)		
00000000G	9F	0C	FB	00458	CALLS	#12, @#SYSSQIOW		3101
	6A	66	D1	0045F	CMPL	(D), (R10)		
		6A	1F	00462	BLSSU	45\$		
		OC	A6	D5	TSTL	12(D)		3107
		OB	13	00467	BEQL	41\$		

0044	8F	00	00000000G	7E 00	08	A6 7D 00469	MOVQ	8(D), -(SP)	3109
					28	02 FB 0046D	CALLS	#2, FREE_VM	3110
					28	A6 D5 00474 41\$:	TSTL	40(D)	3112
					50	0E 13 00477	BEQL	42\$	3113
					30	A6 DD 00479	PUSHL	40(D)	3115
					30	8F 9A 0047C	MOVZBL	#80, -(SP)	3120
					02	FB 00480	CALLS	#2, FREE_VM	3121
					30	A6 D5 00487 42\$:	TSTL	48(D)	3122
					50	0E 13 0048A	BEQL	43\$	3123
					22	A6 DD 0048C	PUSHL	48(D)	3128
					02	8F 9A 0048F	MOVZBL	#80, -(SP)	3131
					00	FB 00493	CALLS	#2, FREE_VM	3132
					66	A6 90 0049A 43\$:	MOVB	34(D), DIR_STRING	3133
					3C	00 2C 004A2	MOVCS	#0, (SP), #0, #68, (D)	3141
					76	004A9	SUBL2	#60, D	3142
					EF	7E 004AD	MOVAQ	-(D), DIR_SP	3143
					12	97 004B4	DECB	DIR_LEVELS	3146
					03	12 004BA	BNEQ	45\$	3154
					26	A8 94 004BC 44\$:	CLRB	3(N)	3156
					24	A8 D4 004BF	CLRL	38(N)	3159
					2C	A8 B4 004C2	CLRW	36(N)	3160
					2A	A8 D4 004C5	CLRL	44(N)	3166
					03B2	A8 B4 004C8	CLRW	42(N)	3171
					66	31 004CB	BRW	87\$	3179
					04	D0 004CE 45\$:	MOVL	(D), R	3180
					01	D0 004D1	MOVL	4(D), V	3184
					012B	E5 004D5	BBCC	#1, DIR_FLAGS, 46\$	3191
					EF	31 004DD	BRW	63\$	3192
					5B	91 004E0 46\$:	CMPB	DIR_STRUCLEV, #1	3193
					57	12 004E7	BNEQ	54\$	3196
					06	D5 004E9 47\$:	TSTL	R	3199
					05	12 004EB	BNEQ	48\$	3200
					10	A6 D0 004ED	MOVL	12(D), R	3204
					59	11 004F1	BRB	49\$	3209
					57	C0 004F3 48\$:	ADDL2	#16, R	3216
					6F	D6 004F6	INCL	V	3219
					67	D1 004F8 49\$:	CMPL	R, 16(D)	3224
					E7	1E 004FC	BGEQU	57\$	3229
					42	B5 004FE	TSTW	(R)	3234
					EF	13 00500	BEQL	47\$	3239
					34	A6 B4 00502	CLRW	66(D)	3244
					08	EF D5 00505	TSTL	DIR_SEL_LATEST	3249
					51	14 0050B	BGTR	53\$	3254
					50	A6 C1 0050D	ADDL3	8(D), 12(D), R1	3259
					06	10 C2 00513	SUBL2	#16, R1	3264
					0A	A6 D0 00516	MOVL	12(D), P	3269
					0E	20 11 0051A	BRB	52\$	3274
					42	A0 D1 0051C 50\$:	CMPL	6(P), 6(R)	3279
					50	16 12 00521	BNEQ	51\$	3284
					08	A0 D1 00523	CMPL	10(P), 10(R)	3289
					08	0F 12 00528	BNEQ	51\$	3294
					08	A0 B1 0052A	CMPW	14(P), 14(R)	3299
					08	1B 0052F	BLEQU	51\$	3304
					08	8F B0 00531	MOVW	#-32768, 66(D)	3309
					08	11 00537	BRB	53\$	3314
					10	C0 00539 51\$:	ADDL2	#16, P	3319



	51		50	D1	0053C	52\$:	CMPL	P, R1		
			DB	1B	0053F		BLEQU	50\$		
			00C0	31	00541	53\$:	BRW	62\$		
			57	D5	00544	54\$:	TSTL	R		3219
			06	12	00546		BNEQ	55\$		
	57	0C	A6	D0	00548		MOVL	12(D), R		3221
			1B	11	0054C		BRB	56\$		
			59	D5	0054E	55\$:	TSTL	V		3223
			17	13	00550		BEQL	56\$		
	50		67	3C	00552		MOVZWL	(R), R0		3226
	50	02	A740	9E	00555		MOVAB	2(R)[R0], NEXT_RECORD		
	59		08	C0	0055A		ADDL2	#8, V		3227
	51	F8	A0	9E	0055D		MOVAB	-8(R0), R1		3228
	51		59	D1	00561		CMPL	V, R1		
			DB	1B	00564		BLEQU	53\$		
	57		50	D0	00566		MOVL	NEXT_RECORD, R		3229
10	A6		57	D1	00569	56\$:	CMPL	R, 16(D)		3235
			D2	1E	0056D	57\$:	BGEQU	53\$		
	51		67	3C	0056F		MOVZWL	(R), R1		3238
FFFF	8F		51	B1	00572		CMPL	R1, #65535		
			17	12	00577		BNEQ	58\$		
52	57	0C	A6	C3	00579		SUBL3	12(D), R, R2		3244
	52	01FF	8F	AA	0057E		BICW2	#511, R2		
	52	0C	A6	C0	00583		ADDL2	12(D), R2		
	57	0200	C2	9E	00587		MOVAB	512(R2), R		
			59	D4	0058C		CLRL	V		3245
			B4	11	0058E		BRB	54\$		3238
	50	02	A741	9E	00590	58\$:	MOVAB	2(R)[R1], NEXT_RECORD		3253
52	57	0C	A6	C3	00595		SUBL3	12(D), R, R2		3257
	52	01FF	8F	AA	0059A		BICW2	#511, R2		
	52	0C	A6	C0	0059F		ADDL2	12(D), R2		
	52	0200	C2	9E	005A3		MOVAB	512(R2), R2		
	52		50	D1	005A8		CMPL	NEXT_RECORD, R2		
			23	1E	005AB		BGEQU	59\$		
	20		51	E8	005AD		BLBS	R1, 59\$		3259
	0E		51	B1	005B0		CMPL	R1, #14		3260
			1B	1F	005B3		BLSSU	59\$		
	50	05	A7	9A	005B5		MOVZBL	5(R), R0		3266
	50	07	A047	9E	005B9		MOVAB	7(R0)[R], R0		
59	50		01	CB	005BE		BICL3	#1, R0, V		
	07	04	A7	93	005C2		BITB	4(R), #7		3267
			08	12	005C6		BNEQ	59\$		
	52		08	C2	005C8		SUBL2	#8, R2		3268
	52		59	D1	005CB		CMPL	V, R2		
			34	1F	005CE		BLSSU	62\$		
		00000000'	EF	95	005D0	59\$:	TSTB	DIR_STRING		3281
			07	12	005D6		BNEQ	60\$		
	50	EAA0	CF	9E	005D8		MOVAB	MFD, R0		
			07	11	005DD		BRB	61\$		
	50	00000000'	EF	9E	005DF	60\$:	MOVAB	DIR_STRING, R0		
		00000000'	50	DD	005E6	61\$:	PUSHL	R0		
			EF	DD	005E8		PUSHL	DIR_DEV_DESC		3280
		00000000G	02	DD	005EE		PUSHL	#2		3277
			8F	DD	005F0		PUSHL	#BACKUP\$ BADDIR		
00000000G	00		04	FB	005F6		CALLS	#4, LIB\$SIGNAL		
	57	10	A6	D0	005FD		MOVL	16(D), R		3286
		18	A6	D4	00601		CLRL	24(D)		3287

		04	66	57	D0	00604	62\$:	MOVL	R, (D)	3297	
		10	A6	59	D0	00607		MOVL	V, 4(D)	3298	
				57	D1	0060B	63\$:	CMPL	R, 16(D)	3304	
				03	1F	0060F		BLSSU	64\$		
				FA37	31	00611		BRW	2\$		
		5B	24	A8	9E	00614	64\$:	MOVAB	36(N), R11	3340	
		02	00000000'	EF	91	00618		CMPB	DIR_STRUCLEV, #2	3317	
				48	12	0061F		BNEQ	67\$		
				42	A6	B7	00621	DECW	66(D)	3320	
		55	05	A7	9A	00624		MOVZBL	5(R), R5	3322	
		54	2C	A6	9E	00628		MOVAB	44(D), R4		
		55		64	B1	0062C		CMPW	(R4), R5		
				08	12	0062F		BNEQ	65\$		
04	B4	06	A7	55	29	00631		CMP3	R5, 6(R), @4(R4)	3325	
				0C	13	00637		BEQL	66\$		
				42	A6	B4	00639	65\$:	CLRW	66(D)	3330
					55	B0	0063C		MOVW	R5, (R4)	3331
04	B4	06	A7	55	28	0063F		MOV3	R5, 6(R), @4(R4)	3335	
				05	A7	9E	00645	66\$:	MOVAB	5(R), NAME	3337
					69	32	00649		CVTTL	(V), VERSION	3338
		00000000'	EF	02	A7	B0	0064C		MOVW	2(R), DIR_VERLIMIT	3339
				02	A9	B0	00654		MOVW	2(V), (R1T)	3340
		26	A8	04	A9	D0	00658		MOVL	4(V), 38(N)	3341
				28	A8	95	0065D		TSTB	40(N)	3343
					39	12	00660		BNEQ	69\$	
		28	A8	20	A6	90	00662		MOVB	32(D), 40(N)	3344
					32	11	00667		BRB	69\$	3317
				E5	AD	9F	00669	67\$:	PUSHAB	FILE_NAME+1	3348
					57	DD	0066C		PUSHL	R	
E5	AD	00000000G	00	02	FB	0066E		CALLS	#2, MAKE_STRING		
			50	3B	3A	00675		LOCC	#59, T, FILE_NAME+1	3349	
				02	12	0067A		BNEQ	68\$		
					51	D4	0067C		CLRL	R1	
E4	AD		50	E5	AD	9E	0067E	68\$:	MOVAB	FILE_NAME+1, R0	
			51		50	83	00682		SUBB3	R0, R1, FILE_NAME	
			5A	E4	AD	9E	00687		MOVAB	FILE_NAME, NAME	3350
			6E	0E	A7	32	0068B		CVTTL	14(R), VERSION	3351
			6B		67	B0	0068F		MOVW	(R), (R11)	3352
		26	A8	02	A7	B0	00692		MOVW	2(R), 38(N)	3353
		28	A8		01	B0	00697		MOVW	#1, 40(N)	3354
		F8	AD	02	A8	9A	0069B	69\$:	MOVZBL	2(N), RSA_DESC	3360
		FC	AD	04	A8	D0	006A0		MOVL	4(N), RSA_DESC+4	3361
					6E	DD	006A5		PUSHL	VERSION	3369
					5A	DD	006A7		PUSHL	NAME	
			00000000'	EF	95	006A9		TSTB	DIR_STRING		
				07	12	006AF		BNEQ	70\$		
		50	E9C7	CF	9E	006B1		MOVAB	MFD, R0		
				07	11	006B6		BRB	71\$		
		50	00000000'	EF	9E	006B8	70\$:	MOVAB	DIR_STRING, R0		
				50	DD	006BF	71\$:	PUSHL	R0		
			00000000'	EF	DD	006C1		PUSHL	DIR_DEV_DESC		
				F8	AD	9F	006C7		PUSHAB	RSA_DESC	
				F8	AD	9F	006CA		PUSHAB	RSA_DESC	
				F927	CF	9F	006CD		PUSHAB	P.AAK	
		00000000G	00	07	FB	006D1		CALLS	#7, SYSSFAO		
		03	A8	AD	90	006D8		MOVB	RSA_DESC, 3(N)	3370	
				58	DD	006DD		PUSHL	N	3371	



00000000G	00	01	FB	006DF	CALLS	#1, INIT_NAMEBLOCK	...	3376
	50	A6	9E	006E6	MOVAB	28(D), R0	...	
2A	A8	60	D0	006EA	MOVL	(R0), 42(N)	...	3378
2E	A8	A0	B0	006EE	MOVW	4(R0), 46(N)	...	3383
	00000000'	EF	94	006F3	CLRB	DIR_STATUS	...	3385
	08	EF	91	006F9	CMPB	DIR_LEVELS, #8	...	
		1F	1A	00700	BGTRU	72\$	...	
	04	A8	91	00702	CMPB	60(N), #4	...	3386
		19	12	00706	BNEQ	72\$	...	
5249442E	8F	B8	D1	00708	CMPL	@80(N), #1380533294	...	3387
		0F	12	00710	BNEQ	72\$	...	
	01	6E	D1	00712	CMPL	VERSION, #1	...	3389
		0A	12	00715	BNEQ	72\$	...	
	04	6B	B1	00717	CMPW	(R11), #4	...	3390
		08	12	0071A	BNEQ	73\$	...	
		29	A8	95	TSTB	41(N)	...	3391
		03	12	0071F	BNEQ	73\$	...	
		00F0	31	00721	BRW	81\$	...	
00000000'	EF	01	88	00724	BISB2	#1, DIR_STATUS	...	3401
DC	AD	EF	9A	0072B	MOVZBL	DIR_STRING, DIR_DESC	...	3407
E0	AD	EF	9E	00733	MOVAB	DIR_STRING+1, DIR_DESC+4	...	3408
		AD	D5	0073B	TSTL	DIR_DESC	...	3409
		0F	13	0073E	BEQL	74\$	...	
		AD	D6	00740	INCL	DIR_DESC	...	3412
	50	EF	9E	00743	MOVAB	DIR_STRING, R0	...	3413
DC BD40	BD40	2E	90	0074A	MOVW	#46, @DIR_DESC[R0]	...	
	51	A8	9A	0074F	MOVZBL	59(N), R1	...	3416
	50	EF	9E	00753	MOVAB	DIR_STRING+1, R0	...	3418
DC BD40	4C	B8	51	28	MOVW	R1, @76(N), @DIR_DESC[R0]	...	
	50	A8	9A	00761	MOVZBL	59(N), R0	...	3419
	DC	AD	50	C0	ADDL2	R0, DIR_DESC	...	
	56	A6	9E	00765	MOVAB	68(R6), D	...	3424
	7E	8F	9A	0076D	MOVZBL	#80, -(SP)	...	3429
00000000G	00	01	FB	00771	CALLS	#1, GET_VM	...	
28	A6	50	D0	00778	MOVL	R0, 40(D)	...	
	7E	8F	9A	0077C	MOVZBL	#80, -(SP)	...	3430
00000000G	00	01	FB	00780	CALLS	#1, GET_VM	...	
30	A6	50	D0	00787	MOVL	R0, 48(D)	...	
	05	EF	E9	0078B	BLBC	DIR_FLAGS, 75\$	...	3437
	50	07	D0	00792	MOVL	#7, STATUS	...	3439
		1C	11	00795	BRB	76\$	...	
		EF	9F	00797	PUSHAB	DIR_SEL_NTV	...	3445
		A6	9F	0079D	PUSHAB	56(D)	...	
		A6	9F	007A0	PUSHAB	36(R6)	...	3444
		EF	9F	007A3	PUSHAB	DIR_SEL_DIR	...	3441
		AD	9F	007A9	PUSHAB	DIR_DESC	...	
00000000G	00	05	FB	007AC	CALLS	#5, MATCH_DIRECTORY	...	3445
32	50	01	E1	007B3	BBC	#1, STATUS, 79\$	...	3449
	51	EF	D0	007B7	MOVL	DIR_SP, R1	...	3452
00000000'	EF	04	88	007BE	BISB2	#4, DIR_STATUS	...	
23	A1	01	88	007C5	BISB2	#1, 35(R1)	...	
	0B	50	E9	007C9	BLBC	STATUS, 77\$	...	3453
00000000'	EF	08	88	007CC	BISB2	#8, DIR_STATUS	...	
23	A6	02	88	007D3	BISB2	#2, 35(D)	...	
04	50	02	E1	007D7	BBC	#2, STATUS, 78\$	...	3454
	23	04	88	007DB	BISB2	#4, 35(D)	...	
2A	50	03	E1	007DF	BBC	#3, STATUS, 80\$	...	3455



0044	8F	00	23	A6	08	88	007E3	BISB2	#8, 35(D)	3449
					24	11	007E7	BRB	80\$	3459
					28	A6	DD 007E9	PUSHL	40(D)	
					50	8F	9A 007EC	MOVZBL	#80, -(SP)	
		00000000G		7E	02	FB	007F0	CALLS	#2, FREE_VM	3460
				00	A6	DD	007F7	PUSHL	48(D)	
				7E	8F	9A	007FA	MOVZBL	#80, -(SP)	
		00000000G		00	02	FB	007FE	CALLS	#2, FREE_VM	
				6E	00	2C	00805	MOVCS	#0, (SP), #0, #68, (D)	3461
					66		0080C			
				56	00000000'	EF	D0 0080D	MOVL	DIR_SP, D	3467
				15	00000000'	EF	E9 00814	BLBC	DIR_FLAGS, 84\$	3474
		03	00000000'	EF	03	E0	0081B	BBS	#3, DIR_FLAGS, 83\$	3475
					00B8	31	00823	BRW	92\$	
				04	6B	B1	00826	CMPW	(R11), #4	3476
					F8	12	00829	BNEQ	82\$	
					29	A8	95 0082B	TSTB	41(N)	3477
					F3	12	0082E	BNEQ	82\$	
				01	00000000'	EF	91 00830	CMPB	DIR_STRUCLEV, #1	3490
					13	12	00837	BNEQ	85\$	
				50	00000000'	EF	9A 00839	MOVZBL	DIR_LEVELS, R0	3492
		00000000'		EF40	59	D1	00840	CMPL	V, DIR_SCANLIMIT-4[R0]	
					40	1B	00848	BLEQU	89\$	
					25	11	0084A	BRB	86\$	
				50	A6	9E	0084C	MOVAB	36(D), R0	3494
					60	B5	00850	TSTW	(R0)	
					36	13	00852	BEQL	89\$	
				7E	A6	3C	00854	MOVZWL	56(D), -(SP)	3503
					04	A0	DD 00858	PUSHL	4(R0)	3502
				7E	60	3C	0085B	MOVZWL	(R0), -(SP)	3501
					0C	AE	DD 0085E	PUSHL	VERSION	3500
					01	AA	9F 00861	PUSHAB	1(NAME)	3499
				7E	6A	9A	00864	MOVZBL	(NAME), -(SP)	
		00000000G		00	06	FB	00867	CALLS	#6, TERMINATE_SCAN	
				19	50	E9	0086E	BLBC	R0, 89\$	
		0A	00000000'	EF	02	E1	00871	BBC	#2, DIR_FLAGS, 88\$	3507
		00000000'		EF	02	88	00879	BISB2	#2, DIR_FLAGS	3510
					50	D4	00880	CLRL	R0	3511
						04	00882	RET		
				66	A6	D0	00883	MOVL	16(D), (D)	3517
					18	A6	D4 00887	CLRL	24(D)	3518
				10	66	D1	0088A	CMPL	(D), 16(D)	3525
					5E	1E	0088E	BGEQU	93\$	
		59	23	A6	01	E1	00890	BBC	#1, 35(D), 93\$	3526
				50	00000000'	EF	D0 00895	MOVL	DIR_SEL_LATEST, R0	3527
					08	14	0089C	BGTR	90\$	
				10	00	EC	0089E	CMPV	#0, #16, 66(D), R0	
					48	12	008A4	BNEQ	93\$	
		0A	00000000'	EF	03	E1	008A6	BBC	#3, DIR_FLAGS, 91\$	3528
				04	6B	B1	008AE	CMPW	(R11), #4	3529
					05	12	008B1	BNEQ	91\$	
					29	A8	95 008B3	TSTB	41(N)	3530
					36	13	008B6	BEQL	93\$	
				50	A8	9A	008B8	MOVZBL	3(N), R0	3539
				50	04	A8	C0 008BC	ADDL2	4(N), R0	
		DC	AD	50	4C	A8	C3 008C0	SUBL3	76(N), R0, NTV_DESC	
				E0	AD	4C	A8	MOV	76(N), NTV_DESC+4	3540



FASTSCAN  
V04-000

Fast file scan  
FIND\_NEXT - find next file

C 9  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 84  
(11)

		00000000'	EF	9F	008CB	PUSHAB	DIR_SEL_NTV	: 3541
		DC	AD	9F	008D1	PUSHAB	NTV_DEST	: :
00000000G	00		02	FB	008D4	CALLS	#2, MATCH_FILENAME	: :
	10		50	E9	008DB	BLBC	R0, 93\$	: :
	1D	00000000'	EF	E9	008DE	BLBC	DIR_STATUS, 96\$	: 3544
00000000'	EF		02	88	008E5	BISB2	#2, DIR_STATUS	: 3545
			14	11	008EC	BRB	96\$	: 3543
03 00000000'	EF		03	E0	008EE	BBS	#3, DIR_FLAGS, 95\$	: 3554
			F752	31	008F6	BRW	2\$	: :
	F9	23	A6	E9	008F9	BLBC	35(D), 94\$	: :
F4	23		03	E1	008FD	BBC	#3, 35(D), 94\$	: :
			50	01	00902	MOVL	#1, R0	: 3562
				04	00905	RET		: :

; Routine Size: 2310 bytes, Routine Base: CODE + 0F84

```
: 2469 3563 1 %SBTTL 'FREE DIR_DATA - free directory scan context'
: 2470 3564 1 GLOBAL ROUTINE FREE_DIR_DATA: NOVALUE=
: 2471 3565 1
: 2472 3566 1 !++
: 2473 3567 1
: 2474 3568 1 FUNCTIONAL DESCRIPTION:
: 2475 3569 1 This routine deletes the directory scan context.
: 2476 3570 1
: 2477 3571 1 INPUT PARAMETERS:
: 2478 3572 1 NONE
: 2479 3573 1
: 2480 3574 1 IMPLICIT INPUTS:
: 2481 3575 1 Directory scan context.
: 2482 3576 1
: 2483 3577 1 OUTPUT PARAMETERS:
: 2484 3578 1 NONE
: 2485 3579 1
: 2486 3580 1 IMPLICIT OUTPUTS:
: 2487 3581 1 Directory scan context.
: 2488 3582 1
: 2489 3583 1 ROUTINE VALUE:
: 2490 3584 1 NONE
: 2491 3585 1
: 2492 3586 1 SIDE EFFECTS:
: 2493 3587 1 NONE
: 2494 3588 1
: 2495 3589 1 --
: 2496 3590 1
: 2497 3591 2 BEGIN
: 2498 3592 2
: 2499 3593 2 ! Free any dynamic storage that is currently allocated.
: 2500 3594 2
: 2501 3595 2 INCRA D FROM DIR_STACK TO DIR_STACK+D_K_NLEVELS*D_S_ENTRY-D_S_ENTRY BY D_S_ENTRY DO
: 2502 3596 3 BEGIN
: 2503 3597 3 MAP
: 2504 3598 3 D: REF BBLOCK; ! Pointer to directory stack entry
: 2505 3599 3
: 2506 3600 3 IF .D[D_BUF_ADDR] NEQ 0
: 2507 3601 3 THEN
: 2508 3602 3 FREE_VM(.D[D_BUF_LEN], .D[D_BUF_ADDR]);
: 2509 3603 3 IF .BBLOCK[D_TERM_DESC], DSC$A_POINTER] NEQ 0
: 2510 3604 3 THEN
: 2511 3605 3 FREE_VM(DIR$$NAME, .BBLOCK[D_TERM_DESC], DSC$A_POINTER);
: 2512 3606 3 IF .BBLOCK[D_NAME_DESC], DSC$A_POINTER] NEQ 0
: 2513 3607 3 THEN
: 2514 3608 3 FREE_VM(DIR$$NAME, .BBLOCK[D_NAME_DESC], DSC$A_POINTER);
: 2515 3609 2 END;
: 2516 3610 2
: 2517 3611 2
: 2518 3612 2 ! Reinitialize the impure storage.
: 2519 3613 2
: 2520 3614 2 CH$FILL(0, DIR_END-DIR_BEG, DIR_BEG);
: 2521 3615 1 END;
```



			00FC 00000	.ENTRY	FREE_DIR_DATA, Save R2,R3,R4,R5,R6,R7	: 3564
57	00000000'	EF	9E 00002	MOVAB	DIR_STACK, R7	:
56	00000000G	00	9E 00009	MOVAB	FREE_VM, R6	:
52		67	9E 00010	MOVAB	DIR_STACK, R2	: 3595
53	0220	C7	9E 00013	MOVAB	DIR_STACK+544, R3	:
		2E	11 00018	BRB	5\$	:
	0C	A2	D5 0001A	TSTL	12(D)	: 3600
		07	13 0001D	BEQL	2\$	:
7E	08	A2	7D 0001F	MOVQ	8(D), -(SP)	: 3602
66		02	FB 00023	CALLS	#2, FREE_VM	:
	28	A2	D5 00026	TSTL	40(D)	: 3603
		0A	13 00029	BEQL	3\$	:
	28	A2	DD 0002B	PUSHL	40(D)	: 3605
7E	50	8F	9A 0002E	MOVZBL	#80, -(SP)	:
66		02	FB 00032	CALLS	#2, FREE_VM	:
	30	A2	D5 00035	TSTL	48(D)	: 3606
		0A	13 00038	BEQL	4\$	:
	30	A2	DD 0003A	PUSHL	48(D)	: 3608
7E	50	8F	9A 0003D	MOVZBL	#80, -(SP)	:
66		02	FB 00041	CALLS	#2, FREE_VM	:
52	44	A2	9E 00044	MOVAB	68(R2), D	: 3595
53		52	D1 00048	CML	D, R3	:
		CD	1B 0004B	BLEQU	1\$	:
03CC	8F		00	MOVCS	#0, (SP), #0, #972, DIR_BEG	: 3614
		00	2C 0004D			:
	FEA0	C7	00054	RET		: 3615
			04 00057			:

; Routine Size: 88 bytes, Routine Base: CODE + 188A

FASTSCAN  
V04-000

Fast file scan  
FREE\_DIR\_DATA - free directory scan context

F 9  
15-Sep-1984 23:56:53  
14-Sep-1984 11:53:52

VAX-11 Bliss-32 V4.0-742  
[BACKUP.SRC]FASTSCAN.B32;1

Page 87  
(13)

: 2523  
: 2524  
3616 1 END  
3617 0 ELUDOM

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes
COMMON	2124	NOVEC, WRT, RD, NOEXE, NOSHR, LCL, REL, OVR, NOPIC, ALIGN(2)
CODE	6370	NOVEC, NOWRT, RD, EXE, NOSHR, LCL, REL, CON, NOPIC, ALIGN(2)

Library Statistics

File	Symbols		Pages Mapped	Processing Time
	Total	Loaded Percent		
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	196 1	1000	00:02.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS\$:FASTSCAN/OBJ=OBJ\$:FASTSCAN MSRC\$:FASTSCAN/UPDATE=(ENH\$:FASTSCAN)

: Size: 6262 code + 2232 data bytes  
: Run Time: 01:53.6  
: Elapsed Time: 06:24.4  
: Lines/CPU Min: 1911  
: Lexemes/CPU-Min: 26165  
: Memory Used: 875 pages  
: Compilation Complete



0011 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

